НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Спеціалізовані комп'ютерні системи

на тему: Методи трасування променів у реальному часі

Виконав: студент II курсу, групи _КВ-73мп_____
(шифр групи)

Грушко Юрій Володимирович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент, д.т.н., Клятченко Я.М. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність 123 Комп'ютерна інженерія

Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В.П. Тарасенко

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Грушку Юрію Володимировичу

1. Тема дисертації «Методи трасування променів у реальному часі», науковий керівник дисертації Клятченко Ярослав Михайлович, д.т.н., доцент, затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації «7» грудня 2018 р.

3. Об'єкт дослідження: процес фізично обґрунтованого рендеринга і процес трасування променів.

4. Предмет дослідження: способи трасування променів та методи розрахунку індексу кольору.

5. Перелік завдань, які потрібно розробити:
   - провести аналіз математичних методів трасування променів;
   - дослідити способи обчислення CRI (Color Rendering Index);
   - обґрунтувати вибір методів для обчислення CRI;
   - запропонувати спосіб інтеграції обчислень CRI у PBRE;
   - розробити програмне забезпечення для реалізації методів трасування променів та обчислення CRI.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: презентація.

7. Орієнтовний перелік публікацій:
   - Тези доповіді на ПКМ 2018 «Методи трасування променів як головна технологія фото реалістичного рендеринга»;
   - Тези доповіді на V Міжнародній науково-технічній Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» «Метод трасування променів на базі рейкастингу».

8. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	15.10.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	11.12.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	21.02.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	02.05.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	04.06.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	28.07.2018	
7.	Підготовка матеріалів четвертого розділу магістерської дисертації	27.09.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2018	20.10.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	03.12.2018	

Студент Ю.В. Грушко

Науковий керівник дисертації Я.М. Клятченко

РЕФЕРАТ

Актуальність теми. Актуальною задачею комп’ютерної графіки являється отримання реалістичних зображень, котрі активно користуються попитом в промисловості, ігровій індустрії та кіно. Фотореалістичне зображення характеризується такими ефектами, як м’які тіні, напівтіні, каустика, динамічне розмиття, глибина різкості, нечіткі відбиття, блиск, напівпрозорість. Серед існуючих підходів фотореалістичної візуалізації методи трасування променів являються найбільш точними, оскільки вони базуються на фізичній моделі поширення світла.

Існує багатий спектр різноманітних методів трасування променів, отже з’являється необхідність у вибірці найбільш ефективних точних методів трасування променів, котрі будуть в середній мірі коректно працювати для широкого ряду статичних (в майбутньому і динамічних) сцен, що проходять візуалізацію.

Об’єктом дослідження є процес фізично обґрунтованого рендерингу і процес трасування променів.

Предметом дослідження є способи трасування променів та методи розрахунку індексу передачі кольору.

Мета роботи: дослідження методів PBR (Physical Based Rendering), їх одночасного використання для отримання максимального ефекту реалізму; оцінка здатності джерела світла виявляти всі частоти його кольорового спектру у порівнянні з контрольним світлом.

Наукова новизна, а точніше – інноваційне рішення полягає в тому, що розроблений рушій реалізує обчислення індексу передачі кольору (CRI - Color Rendering Index) з високим рівнем точності відносно очікуваних значень контрольних джерел світла.

Практична цінність проведених досліджень полягає у розробці нового PBRE, який для рендерингу сцен використовує емпіричні моделі освітлення; реалізовані такі моделі BRDF, як Ламберта, Орена Найара, Торренса Спарроу, дзеркального відбиття, дзеркального пропускання і виміряного BRDF. Реалізована підтримка декількох технік трасування променів: трасування

Уайтеда і трасування шляху. Розраховуються кольори з використанням спектральних даних і колірний простір CIE XYZ в сценах PBR для досягнення високої передачі кольору. TTFD також підтримує обчислення індексу передачі кольору (CRI – Color Rendering Index). Цей показник описує здатність джерела світла точно відображати всі частоти його колірного спектра в порівнянні з ідеальним еталонним світлом аналогічного типу.

Структура та обсяг роботи. Магістерський дипломний проект складається зі вступу, чотирьох розділів та висновків.

У *вступі* подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи.

У *першому розділі* розглянуто принципи колориметрії та радіометрії. Вони складають основу деяких основних ключових особливостей TTFD. Зокрема, розрахунок кольору і методи освітлення / затінення, реалізовані в TTFD, використовують поняття, представлені даному розділі.

У *другому розділі* розглянуто трасування променів: фотореалістичний рендеринг (візуалізація). Коротка класифікація алгоритмів трасування променів. Вирішення рівняння рендеринга.

У *третьому розділі* наведено особливості реалізації розробленої системи.

У *четвертому розділі* представлено підходи до тестування системи в цілому та окремих модулів.

У *висновках* представлені результати проведеної роботи.

Робота представлена на 116 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: фізичний рендеринг (PBR), трасування променів, індекс передачі кольору, емпіричні моделі освітлення, модель Уайтеда, трасування шляху.

РЕФЕРАТ

Актуальность темы. Актуальной задачей компьютерной графики является получение реалистичных изображений, которые активно пользуются спросом в промышленности, игровой индустрии и кино. Фотореалистичное изображение характеризуется такими эффектами, как мягкие тени, полутени, каустика, динамическое размытие, глубина резкости, нечеткие отражение, блеск, полупрозрачность. Среди существующих подходов фотореалистичной визуализации методы трассировки лучей являются наиболее точными, поскольку они базируются на физической модели распространения света.

Существует богатый спектр различных методов трассировки лучей, следовательно появляется необходимость в выборке наиболее эффективных точных методов трассировки лучей, которые будут в средней степени правильно работать для широкого ряда статических (в будущем и динамических) сцен, проходят визуализацию.

Объектом исследования является процесс физически обоснованного рендеринга и процесс трассировки лучей.

Предметом исследования являются способы трассировки лучей и методы расчета индекса цветопередачи.

Цель работы: исследование методов PBR (Physical Based Rendering), их одновременного использования для получения максимального эффекта реализма; оценка способности источника света выявлять все частоты его цветового спектра по сравнению с контрольным светом.

Научная новизна, а точнее - инновационное решение, заключается в том, что разработан двигатель реализует вычисления индекса цветопередачи (CRI - Color Rendering Index) с высокой степенью точности относительно ожидаемых значений контрольных источников света.

Практическая ценность проведенных исследований состоит в разработке нового PBRE, который для рендеринга сцен использует эмпирические модели освещения; реализованы такие модели BRDF, как Ламберта, Орена Найара, Торренса Спарроу, зеркального отражения, зеркального пропускания и измеренного BRDF. Реализована поддержка нескольких техник трассировки

лучей: трассировки Уайтеда и трассировки пути. Рассчитываются цвета с использованием спектральных данных и цветовое пространство CIE XYZ в сценах PBR для достижения высокой цветопередачи. TTFD также поддерживает вычисления индекса цветопередачи (CRI - Color Rendering Index). Этот показатель описывает способность источника света точно отражать все частоты его цветового спектра по сравнению с идеальным эталонным светом аналогичного типа.

Структура и объем работы. Магистерский дипломный проект состоит из введения, четырех глав и выводов.

Во введении представлена общая характеристика работы, произведена оценка современного состояния проблемы, обоснована актуальность направления исследований, сформулированы цели и задачи исследований, показано научную новизну полученных результатов и практическую ценность работы.

В первом разделе рассмотрены принципы колориметрии и радиометрии. Они составляют основу некоторых основных ключевых особенностей TTFD. В частности, расчет цвета и методы освещения / затенения, реализованные в TTFD, используют понятие, представленные данном разделе.

Во втором разделе рассмотрены трассировки лучей: фотореалистичный рендеринг (визуализация). Краткая классификация алгоритмов трассировки лучей. Решение уравнения рендеринга.

В третьем разделе приведены особенности реализации разработанной системы.

В четвертом разделе представлены подходы к тестированию системы в целом и отдельных модулей.

В выводах представлены результаты проведенной работы.

Работа представлена на 116 листах, содержит ссылки на список использованных литературных источников.

Ключевые слова: физический рендеринг (PBR), трассировки лучей, индекс цветопередачи, эмпирические модели освещения, модель Уайтеда, трассировка пути.

ABSTRACT

Relevance of the topic. The actual task of computer graphics is to obtain realistic images that are actively in demand in industry, gaming and film industry. A photorealistic image is characterized by such effects as soft shadows, partial shade, caustic, dynamic blur, depth of field, fuzzy reflection, shine, translucency. Among the existing approaches of photorealistic visualization, ray tracing methods are the most accurate because they are based on a physical model of light propagation.

There is a wide range of different ray-tracing methods, and therefore there is a need to select the most efficient, accurate ray-tracing methods that will, in average, work correctly for a wide range of static (future dynamic) scenes, and are being visualized.

The object of the research is the process of physically sound rendering and the ray tracing process.

The subject of research is the methods of ray tracing and methods for calculating the color rendering index.

Objective: to study the methods of PBR (Physical Based Rendering), their simultaneous use to obtain the maximum effect of realism; assessment of the ability of a light source to detect all the frequencies of its color spectrum compared to the control light.

The scientific novelty, or rather, an innovative solution, is that the engine developed implements the calculations of the color rendering index (CRI - Color Rendering Index) with a high degree of accuracy relative to the expected values of the control light sources.

The practical value of the research is the development of a new PBRE, which employs empirical lighting models for rendering scenes; BRDF models such as Lambert, Oren Nayar, Torrens Sparrow, specular reflection, specular transmission and measured BRDF are implemented. Implemented support for several ray tracing techniques: Traced by Wyted and path tracing. Colors are calculated using spectral data and CIE XYZ color space in PBR scenes to achieve high color rendering. TTFD also supports Color Rendering Index (CRI) calculations. This indicator describes the

ability of a light source to accurately reflect all the frequencies of its color spectrum compared to ideal reference light of a similar type.

Structure and scope of work. Master thesis project consists of introduction, four chapters and conclusions.

The introduction presents a general description of the work, assesses the current state of the problem, substantiates the relevance of the research area, formulates the goals and objectives of the research, shows the scientific novelty of the results and practical value of the work.

The first section discusses the principles of colorimetry and radiometry. They form the basis of some key TTFD key features. In particular, color calculations and lighting / shading methods implemented in TTFD use the concept presented in this section.

The second section deals with ray tracing: photorealistic rendering (visualization). Brief classification of ray tracing algorithms. Solution of the rendering equation.

The third section presents the features of the implementation of the developed system.

The fourth section presents approaches to testing the system as a whole and individual modules.

The findings present the results of this work.

The work is presented on 116 pages, contains links to the list of references used.

Keywords: physically based rendering (PBR), ray tracing, color rendering index, empirical lighting models, Whited model, path tracing.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AABB	Axis Aligned Bounding Box
BRDF	Bidirectional Reflectance Distribution Function
BTDF	Bidirectional Transmittance Distribution Function
CI	Continuous Integration
CAT	Chromatic Adaptation Transform
CCT	Color Correlated Temperature
CIE	Commission Internationale de l'Éclairage
CMF	Color Matching Function
CRI	Color Rendering Index
GCC	Gnu Compiler Collection
GCD	Grand Central Dispatcher
MERL	Mitsubishi Electric Research Laboratories
MVC	Model View Controller
PBR	Physically Based Rendering
RGB	Red Green Blue
SPD	Spectral Power Distribution
UI	User Interface
UWP	Universal Windows Platform
XAML	eXtensible Application Markup Language

СИМВОЛИ

λ	Wavelenght
$E(\lambda)$	Illuminant SPD
$S(\lambda)$	Generic object spectrum
$CMF(\lambda)$	Color matching function
$\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)$	Color matching function component
$V(\lambda)$	Photopic luminous efficiency function
X, Y, Z	Tristimulus value (CIE 1931 color space)
x, y, z	XYZ chromaticity coordinates
u, v	CIE 1960 color space chromaticity coordinates
U^*, V^*, W^*	CIE 1964 color space coordinates
L, a, b	CIE 1976 color space coordinates
$u_{c,i}, v_{c,i}$	Von Kries CAT components
ϕ	Radiant flux
E	Irradiance
L	Radiance
p	Point
ω_i	Incident light direction (vector)
ω_0	Outgoing light direction (vector)
$L_i(p, \omega)$	Incident radiance
$L_0(p, \omega)$	Exitant radiance
$L_e(p, \omega_0)$	Emitted radiance
$f_r(p, \omega_i, \omega_0)$	Bidirectional Reflectance Distribution Function
$f_t(p, \omega_i, \omega_0)$	Bidirectional Trasmittance Distribution Function
$\omega_i(\theta_i, \phi_i)$	Spherical coordinates of incident light direction
$\omega_0(\theta_0, \phi_0)$	Spherical coordinates of outgoing light direction
R	Reflectance spectrum

δ	Dirac delta function
η	Index of refraction
F_r	Fresnel reflection
X	Random variable
$P(x)$	Cumulative Distribution Function
$p(x)$	Probability Density Function
$E_p[f(x)]$	Expected value of a function
F_N	Monte Carlo estimator
R_i	Special CRI
R_a	General CRI

ВСТУП

Фізичний рендеринг (Physically based rendering or PBR, rendering - візуалізація, проявлення, вимальовування, подання) є одним із найбільш технологічних і трендових напрямків у комп'ютерній графіці. PBR використовує фізично правильні моделі освітлення і затінення для оброки світла так, як це відбувається у справжньому світі. Внаслідок того, що те, що можна побачити в програмі з комп'ютерною графікою, такою як відеогра, представляє собою CAD (computer-aided design) медичну модель, визначається тим, як представлене світло і з PBR дає можливість досягнути нового рівня реалізму.

У цій магістерській дисертації був створений новий PBRE (PBR Engine) з нуля для вивчення деяких можливостей PBR та кольорометрії (колориметрії) у комп'ютерній графіці: Test Tracer for Diploma (TTFD).

TTFD – це крос-платформний рушій трасування променів, який підтримує безліч моделей освітлення і затінення на фізичному рівні. Він також повністю на даних спектрального розподілу, щоб досягти максимального рівня точності передачі кольору.

Що таке трасування променів? Трасування променів - це техніка комп'ютерної графіки, яка може генерувати реалістичні зображення, відстежуючи шлях світла в сцені. З визначення легко бачити, що ця техніка добре поєднується з PBR.

Його основним недоліком є те, що для генерації одного кадру потрібна величезна кількість обчислювальної потужності. Фактично, трасування променів зазвичай не є технікою рендеринга в реальному часі.

TTFD підтримує різні типи трасування променів:

• Модель Уайтеда (Turner Whitted)

• Трасування шляху (Path tracing)

Перший - це класична модель трасування променів. Він може відображати такі матеріали, як скло або дзеркало, які важко отримати в стандартному тривимірному режимі реального часу. Другий - це техніка,

здатна відображати сцени з високим рівнем реалізму. Сцени, створені з використанням TTFD з трасуванням шляху покажуть реалістичне фізичне явище, яке важко отримати за допомогою моделі Уайтеда.

Як було сказано раніше, TTFD приділяє велику увагу правильності кольору об'єктів, створених в сцені. Всі обчислення кольору засновані на спектральних даних і значеннях тристимула CIE 1931 XYZ. Таким чином, TTFD здатний точно представляти фізичні феномени, строго пов'язані з кольором, такі як метамеризма (метамерія), які важко отримати при стандартних розрахунках кольору RGB.

TTFD також підтримує обчислення індексу передачі кольору (CRI – Color Rendering Index). Цей показник описує здатність джерела світла точно відображати всі частоти його колірного спектра в порівнянні з ідеальним еталонним світлом аналогічного типу. CRI оцінюється за шкалою від 0 до 100. Чим нижче CRI, тим менш точно відтворюються кольори [1]. Таким чином, TTFD має можливість оцінити, наскільки точно джерело світла показує кольори в сцені.

Завдяки цим функціям TTFD - ідеальний інструмент для промислового освітлення і виробництва. Фактично, його можна використовувати для оцінки якості нового світлового продукту (new light product) з використанням CRI. Дизайнери по освітленню можуть також оцінити точність кольору своїх нових світлових продуктів у візуалізованих сценах з використанням PBR і спектральних даних для отримання найбільш точних результатів. Як крос-платформний рушій, що підтримує пристрої Apple iOS / OS X і ПК з Microsoft Windows 10, дає користувачеві свободу вибору кращої платформи без будь-яких обмежень.

Який рівень розвитку PBR і трасування променів? Як TTFD зрівнюється з іншими рушіями трасування променів з фізичною основою?

PBR і трасування променів вже використовуються в тандемі у виробничому середовищі. Наприклад, дві найвідоміші студії анімації,

студії анімації Уолта Діснея і Pixar, використовують PBR і трасування променів для виробництва своїх анімаційних фільмів.

Зокрема, студія анімації Уолта Діснея використовує власний PBR трасувальник шляху для створення своїх анімаційних фільмів з 2014 року: Hyperion [2][3]. Очевидно, що цей механізм рендеринга добре оптимізований і впроваджує нові інноваційні методи [2]:

Hyperion (рис. 1)обробляє кілька мільйонів світлових променів за раз, сортуючи і об'єднуючи їх разом відповідно до їх напрямку. Коли промені групуються таким чином, то велика кількість променів в пучку потрапляють в один і той же об'єкт в тій же області простору. Ця схожість променів дозволяє нам - і комп'ютерам - оптимізувати обчислення для ударів з об'єктами.

Рисунок 1 – «Великий герой 6», перший анімаційний фільм Walt Disney зроблений на Hyperion

Pixar також поліпшила механізм рендеринга, RenderMan. Фактично, тепер він включає нову структуру рендеринга RIS [4], засновану на трасуванні променів і оптимізовану для PBR. Результати роботи можна побачити на рисунку 2.

На даний момент TTFD не може зрівнятися з цими рушіями, оскільки вони є результатом багаторічної розробки командами двох анімаційних студій.

TTFD не може зрівнятися з іншим рушієм рендеринга, орієнтованим на продуктивність. Навіть якщо, як описано вище, трасування променів зазвичай не є технологією в режимі реального часу, в результаті досліджень в комп'ютерній графіці був створений певний рушій трасування променів майже в реальному часі. Один з них - Brigade, вражаючий рушій трасування в реальному часі [5]. З іншого боку, TTFD займає багато годин, щоб зробити одне зображення сцени із середнім ступенем
складності.

Рисунок 2 – «Університет монстрів», перший мультфільм Pixar зроблений за допомогою RIS

TTFD можна було б більш точно порівняти з іншим рушієм, доступним на ринку: LuxRender [6], рушій рендеринга з відкритим вихідним кодом. Цей рушій, як і TTFD, використовує фізично правильні моделі освітлення і матеріалів і володіє багатьма іншими функціями: відображення тонів, черговість і рендеринг мережі, кілька моделей камер, різні ефекти, такі як глибина різкості і розмитість зображення. Він поширюється як крос-платформний додаток, доступний для операційних систем Microsoft Windows, Apple OS X і Linux.

Фактично, LuxRender, як і TTFD, народжується з вивчення найвідомішого механізму PBR з відкритим вихідним кодом: PBRT від Мета Фара і Грега Гампфрейза [7][8], демонстрація на рисунку 3. Однією з ключових особливостей TTFD, що відрізняють його від інших рушіїв, є CRI: жоден з раніше згаданих двигунів не підтримує його обчислення.

Більш детально про основні особливості TTFD:

• рендеринг сцен з використанням PBR і емпіричних моделей освітлення. Таким чином, можна порівняти фізично створені спектральні сцени зі стандартними емпіричними RGB сценами. Зокрема, для PBR TTFD реалізує наступні моделі BRDF: Ламберта, Орена-Найара [9], Торренса-Спарроу [10], дзеркального відбиття, дзеркального пропускання і виміряної BRDF [11][12];

- підтримка декількох технік трасування променів:
    - Трасування Уайтеда, створена Тернером Уайтедом в 1979 році [13], метод, в якому простежуються різні види променів: відбиті промені для дзеркальних поверхонь, заломлені промені для пропускних поверхонь (наприклад, скла) і тіньових променів. Він реалізується з використанням PBR і емпіричних моделей;
    - Трасування шляху – метод, представлений Дж. Т. Каджая в 1986 році [14]. Він використовує метод інтеграції Монте-Карло, щоб знайти чисельне рішення раніше сформульованого рівняння рендеринга. Він реалізований з використанням тільки моделей PBR;
- Обчислення кольору з використанням спектральних даних і колірний простір CIE XYZ в сценах PBR для досягнення високої передачі кольору;
- розрахунок CRI для джерел світла, що містяться в сценах PBR, з використанням двох методів: метод тестових зразків [15] і R96a [16];

- крос-платформна підтримка: TTFD є нативним додатком для будь-якого пристрою Apple iPad, починаючи з iPad 2 (iOS 9.0), будь-якого комп'ютера Apple, який підтримує OS X El Capitan 10.11 і будь-який ПК, що підтримує Windows 10.

У наступних розділах буде показано короткий вступ в основні поняття, пов'язані з фізичним обґрунтуванням, радіометрією, колориметрією і трасуванням променів. Потім буде представлено детальний опис TTFD, в якому основна увага буде приділена:

• загальна реалізація та архітектура;

• основні функції;

• серію тематичних досліджень, що показують деякі зроблені сцени, отримані з TTFD, і розрахунок CRI для різних типів світла.

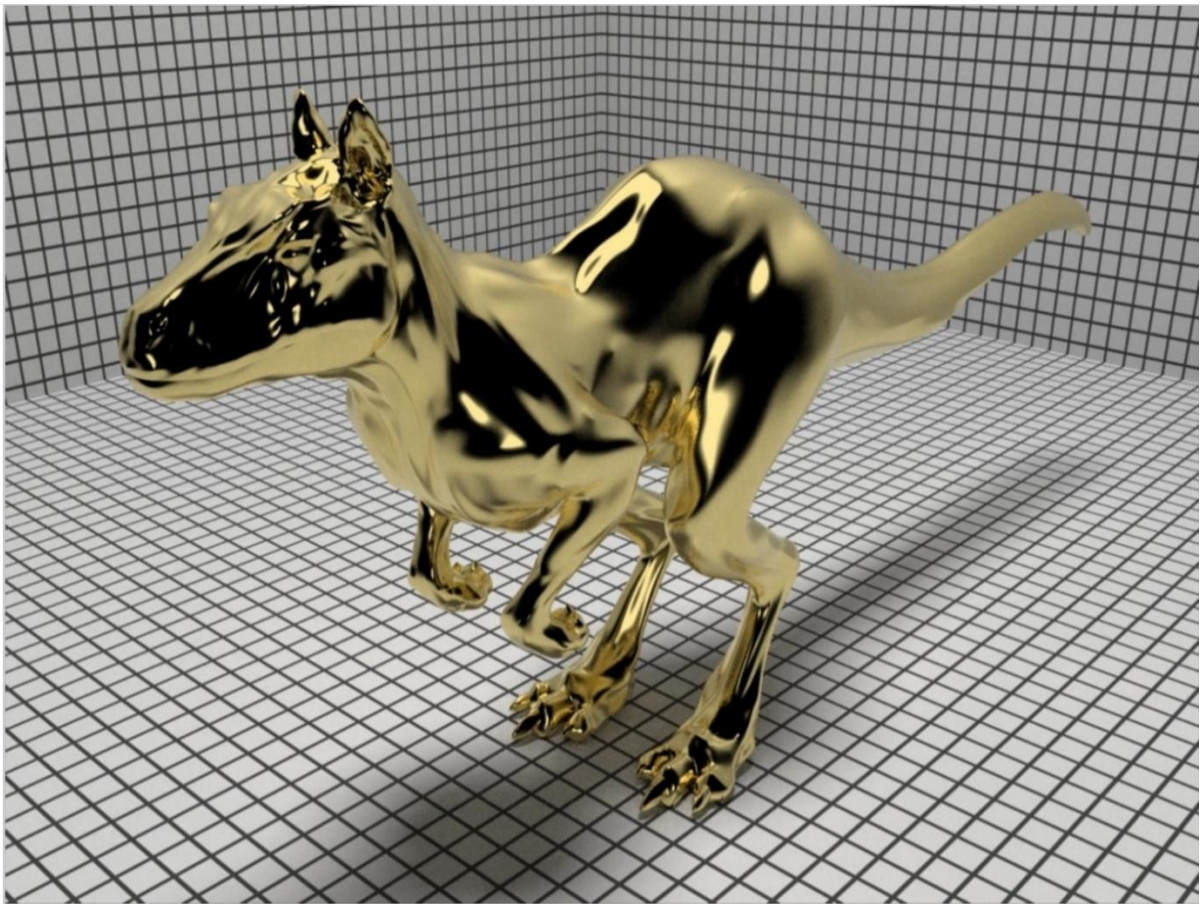

Рисунок 3 – Приклад сцени, яку відрендерили за допомогою PBRT

РОЗДІЛ 1. ПРИНЦИПИ КОЛОРИМЕТРІЇ ТА РАДІОМЕТРІЇ

1.1 Колориметрія

Колір - це сприйняття, тісно пов'язане зі спостерігачем, який намагається описати його. На те, як це сприймається, може впливати безліч факторів: вік, втома, фізіологічні чинники.

Як відчуття, колір складно описати та виміряти. Ось чому була створена колориметрія.

Колориметрія - це наука, яка намагається фізично описати сприйняття кольору людиною. Основним авторитетом в цій області є Commission International de l'Eclairage (CIE). Вона була створена в 1913 році і з тих пір відповідає за визначення і специфікацію нового стандарту колориметрії через свої публікації. Одна з її основних цілей - визначити організацію кольорів: кольорові простори. У 1931 році CIE визначив основу колориметрії використовуючи значення тристимула: колірний простір CIE XYZ.

1.1.1 Значення тристимула: кольоровий простір CIE 1931 XYZ

Як було описано вище, колір це сприйняття.

Зокрема, людські очі (рисунок 4) чутливі до невеликої частини всього спектра електромагнітного випромінювання в діапазоні довжин хвиль від 380 до 780 нм (рисунок 5). Око має три різних типи фоторецепторів у сітківці, звані конусоподібними клітинами, з чутливим піком в різних діапазонах спектру: короткохвильовий конус S, конус середньої довжини хвилі M і довгохвильовий конус L [17]. Діапазон вище описаного спектра електромагнітного випромінювання зазвичай називається видимим світлом.

Колір може бути представлений з використанням тільки трьох значень. Це перший з трьох принципів, описаних в законах Германа Грассмана [18]:

- Увесь колір може бути заданий з використанням трьох різних незалежних змінних;

- Стимули з різним спектральним складом можуть давати однакове співпадіння кольору. Це називається метамеризмом;
- Якщо змінюється один компонент кольору, то він повинен змінюватися відповідним чином.

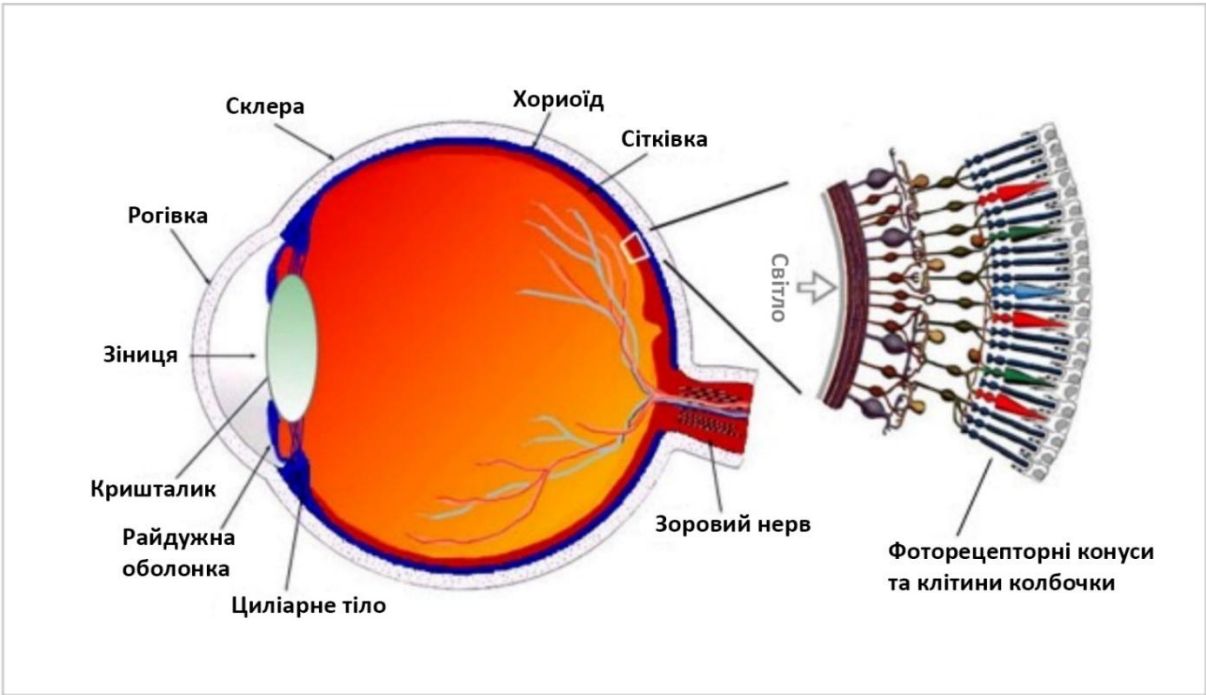

Рисунок 4 – Анатомія ока та фоторецепторних конусів [19]

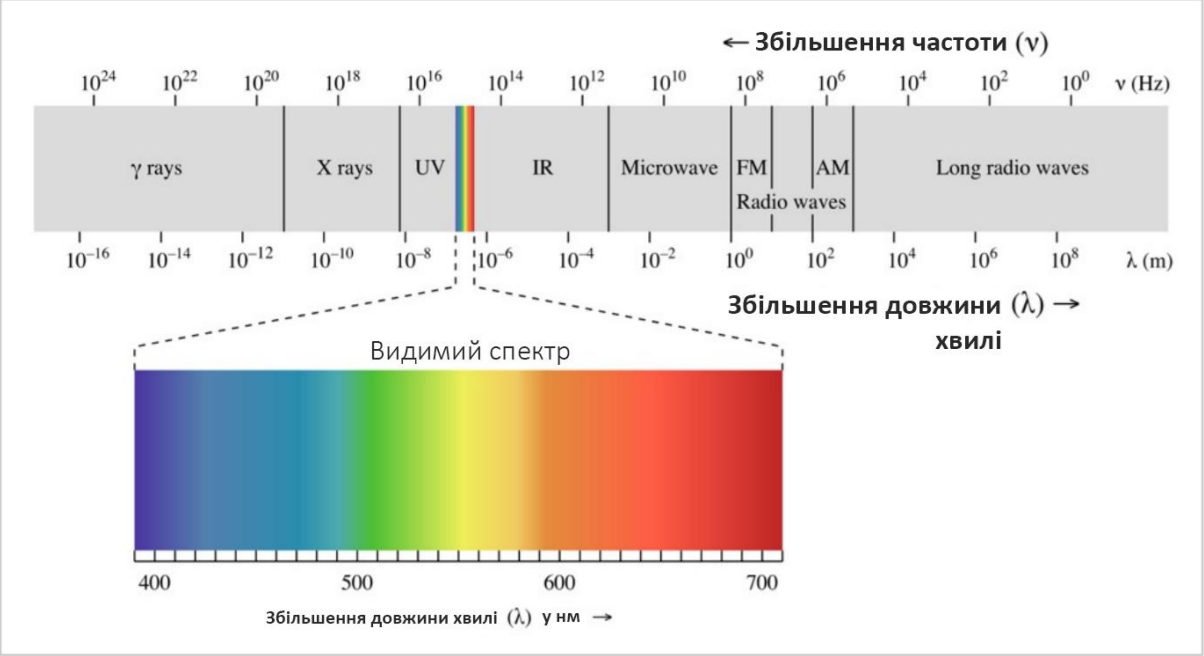

Рисунок 5 – Видиме світло: мала частина спектру електромагнітного випромінення

CIE використовувала закони спектрального розподілу потужності (Spectral power distribution, SPD) світла і спектр об’єктів для визначення значень тристимула CIE і колірного простору CIE XYZ, щоб ідентифікувати колір чисельно. Вони визначаються як інтеграція продукту освітлення SPD $E(\lambda)$, спектру об'єктів $S(\lambda)$ і функцій узгодження кольорів $CMF(\lambda) = \{\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)\}$ помножених на нормуючий фактор (тобто дати вибрати 1 для яскравості Y) [20].

$$X = \frac{1}{\int \bar{y}(\lambda)E(\lambda)d\lambda} \int \bar{x}(\lambda)E(\lambda)d\lambda \quad (1)$$

$$Y = \frac{1}{\int \bar{y}(\lambda)E(\lambda)d\lambda} \int \bar{y}(\lambda)E(\lambda)d\lambda \quad (2)$$

$$Z = \frac{1}{\int \bar{y}(\lambda)E(\lambda)d\lambda} \int \bar{z}(\lambda)E(\lambda)d\lambda \quad (3)$$

Спектральний розподіл потужності являє собою функцію розподілу довжини хвилі, яка описує кількість світла на кожній з довжин.

Спектр об'єктів можна отримати з використанням спеціальних інструментів, наприклад спектрофотометра. Функції узгодження кольорів також називаються стандартними спостерігачами CIE. Це числове представлення хроматичної реакції людини-спостерігача. Ці функції отримані за допомогою експерименту: спостерігач повинен відповідати тестовому зразку з трьома відповідними стимулами $r(\lambda), g(\lambda)$ і $b(\lambda)$. Отримані значення потім масштабуються за допомогою функції фотоплівкової світлової ефективності $V(\lambda)$ визнаною CIE в 1924 році [20].

Насправді доступні два типи стандартного спостерігача:

- 2°, також званий стандартним спостерігачем CIE 1931 який являє собою спостерігача-людини з полем зору 2°, з відповідними стимулами при 700, 546,1 і 435,8 нм;
- 10 °, стандартний спостерігач, також званий CIE 1964 Standard Observer, який являє собою спостерігача-людини з полем зору 10 °, з відповідними стимулами при 645,2, 526,3 і 444,4 нм.

Освітлений SPD зазвичай описує спектральний склад джерела світла. Доступні кілька джерел світла SPD, які будуть обговорюватися у пункті 1.1.5.

Спектр і SPD зазвичай подаються у вигляді набору зразків по діапазону довжин хвиль видимого світла з певною частотою дискретизації. Найбільш часто використовувана частота дискретизації становить 5 нм. Таким чином, інтеграли у формулах (1), (2), (3) можна розрахувати з сумою Рімана:

$$X = \frac{1}{\sum \bar{y}(\lambda)E(\lambda)\Delta\lambda}\sum \bar{x}(\lambda)\,E(\lambda)S(\lambda)\Delta\lambda \tag{4}$$

$$Y = \frac{1}{\sum \bar{y}(\lambda)E(\lambda)\Delta\lambda}\sum \bar{y}(\lambda)\,E(\lambda)S(\lambda)\Delta\lambda \tag{5}$$

$$Z = \frac{1}{\sum \bar{y}(\lambda)E(\lambda)\Delta\lambda}\sum \bar{z}(\lambda)\,E(\lambda)S(\lambda)\Delta\lambda \tag{6}$$

де $\Delta\lambda$ – інтервал вибірки. Коли він постійний, то відповідає:

$$\Delta\lambda = \frac{\lambda_t}{(n-1)} \tag{7}$$

де λ_t – повний діапазон спектра, а n – кількість вибірок.

Хроматичні координати CIE 1931 XYZ

Колір має три основні властивості: відтінок, кольоровість і яскравість.

Відтінок - це те, як сприймається колір: червоний, оранжевий, зелений, синій і т.д. Кольоровість (іноді звана також насиченням) описує наскільки близько колір до сірого або чистого відтінку. Яскравість пов'язана зі ступенем світлоти кольору.

Іноді корисно представляти колір незалежно від його яскравості, фокусуючи увагу тільки на властивостях відтінку і кольоровості.

Колір в CIE XYZ (рисунок 6) можна вказати тільки для цих двох властивостей, використовуючи координати кольоровості, які визначаються як нормалізація значень тристимула:

$$x = \frac{X}{X + Y + Z} \tag{8}$$

$$y = \frac{Y}{X + Y + Z} \tag{9}$$

$$z = \frac{Z}{X + Y + Z} \tag{10}$$

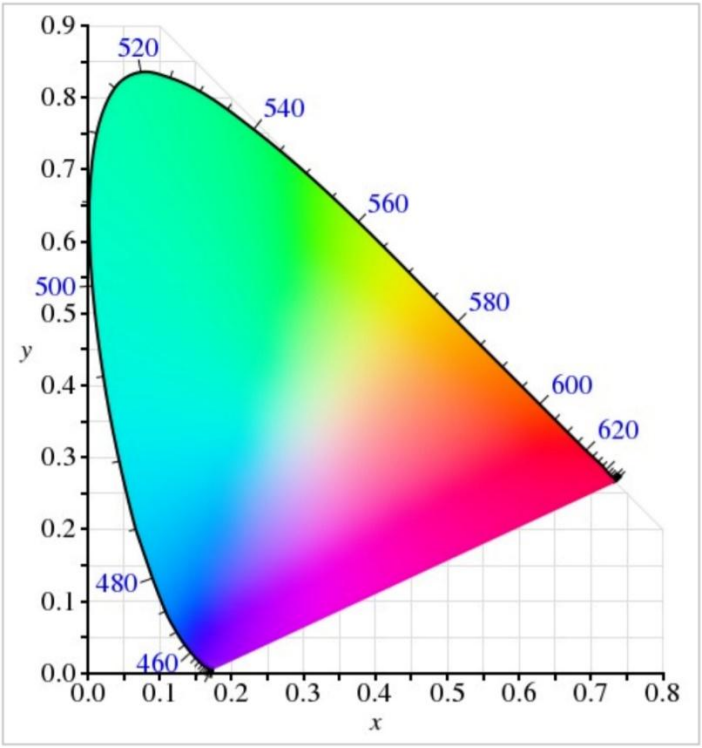

Рисунок 6 – Діаграма хроматичності CIE 1931 XYZ

1.1.2 Кольоровий простір CIE 1960 UCS

Кольорова палітра CIE 1960 також звана єдиним кольоровим простором, являє собою простір кольору, який фокусується, подібно координатами кольору CIE XYZ, на колірних відтінках і хроматичних властивостях кольору. Це було запропоновано Девідом Л. МакАдамом в 1937 році [21] і воно широко використовується для розрахунку корельованої колірної температури, описаної у пункті 1.1.5.

Формула, яка використовується для перетворення значень тристимула CIE XYZ в CIE 1960 UCS, наступна:

$$u = \frac{4X}{X + 15Y + 3Z} \tag{11}$$

$$v = \frac{6X}{X + 15Y + 3Z} \quad (12)$$

Формула для перетворення з хроматичних координат CIE XYZ в CIE 1960 UCS:

$$u = \frac{4x}{12y - 2x + 3} \quad (13)$$

$$v = \frac{6x}{12y - 2x + 3} \quad (14)$$

1.1.3 Колірний простір CIE 1964 UVW

У 1963 році Гюнтер Вішецький створив колірний простір CIE 1964 UVW [22]. Він базується на раніше описаному колірному просторі CIE 1960 UCS. Формули, що використовуються для отримання координат:

$$U^* = 13W^*(u - u_0) \quad (15)$$

$$V^* = 13W^*(v - v_0) \quad (16)$$

$$W^* = 25Y^{1/3} - 17 \quad (17)$$

U^* і V^* є перетворенням хроматичних координат CIE XYZ так, що біла точка з хроматичними координатами CIE XYZ (u_0, v_0) відображається в початок координат. W^* відповідають індексу яскравості.

1.1.4 Колірний простір CIE 1976 L* a* b*

Кольоровий простір CIE 1976 L* a* b* базується на колірному просторі CIE XYZ. L^* визначає світлоту, a^* визначає значення червоного/зеленого і b^* визначає значення жовтого/блакитного [23]. a^* і b^* не мають певного діапазону, в той час як L^* визначається в межах 0 – 100.

Цей колірний простір здатний представляти весь людський кольоровий зір. Фактично, його зазвичай порівнюють з колірним простором CIE XYZ.

Формули, які використовуються для перетворення координат з CIE XYZ в CIE L* a* b* є [24]:

$$L = 116f\left(\frac{Y}{Y_r}\right) - 16 \tag{18}$$

$$a = 500\left(f\left(\frac{X}{X_r}\right) - f\left(\frac{Y}{Y_r}\right)\right) \tag{19}$$

$$b = 200\left(f\left(\frac{X}{X_r}\right) - f\left(\frac{Z}{Z_r}\right)\right) \tag{20}$$

У попередній формулі (X_r, Y_r, Z_r) відповідають значенням триколірного сигналу еталонної білої точки – набір значень, які представляють собою білий колір для конкретного середовища.

$f(a)$ визначається як:

$$f(x) = \begin{cases} \sqrt[3]{a} & a > \varepsilon \\ \dfrac{\kappa a + 16}{116} & a \leq \varepsilon \end{cases} \tag{21}$$

де κ та ε відповідають стандарту CIE наступним значенням:

$$f(x) = \begin{cases} 0.008856 \text{ фактичний стандарт } CIE \\ \dfrac{216}{24389} \text{ намір стандарту } CIE \end{cases} \tag{22}$$

1.1.5 Стандартні джерела світла

Як раніше описано в пункті 1.1.1, джерело світла є специфічним SPD, який може бути пов'язаний з певним типом джерела світла. CIE визначила деякі сімейства стандартних джерел світла. Тут наводяться ті, що використовуються TTFD:

- джерело світла A: воно еквівалентне ламповому освітленню [25]. Його SPD розраховується у відповідності з законом випромінювання Планка:

$$E(\lambda, T) = \frac{c1}{\lambda^5} * \frac{1}{e^{(c2/\lambda T)} - 1} \tag{23}$$

де $c1 = 2\pi hc^2$ і $c2 = \frac{hc}{k}$. Константа, яка використовується у формулі вище – це стала Планка $h = 6.62670 \times 10^{-34}$Дж · с,

швидкість світла $c = 2.99792458 \times 10^8 \frac{\text{м}}{\text{с}}$ і стала Больцмана $k = 1.3806505 \times 10^{-23} \text{Дж} \cdot \text{К}^{-1}$;

- джерело світла D: це сімейство джерел світла є математичним представленням різних фаз денного світла [26]. Найбільш широковживані: D65, яке представляє полудневе денне світло, та D50, яке представляє горизонтальне денне світло.
- Джерело світла F: це сімейство джерел світла включає різні типи флуоресцентного світла [27]. Найбільш важливим джерелом світла цього сімейства є FL4, яке використовується в якості еталону для калібрування CRI.

Однією з корисних властивостей джерел світла є колірна кореляція температури (CCT - Color Correlated Temperature): колірна температура чорного тіла радіатора, ідеальне фізичне тіло, що поглинає все падаюче світло, має майже той же колір, що і джерело світла.

1.1.6 Перевірка кольору Macbeth

План перевірки кольору Macbeth являє собою набір кольорів, який можна використовувати в якості довідкової інформації для досліджень в області колориметрії.

Спочатку перевірка називалася «Колірна карта» і була описана в статті С. С. Маккей, Х. Маркуса і Дж. Г. Девідсона в 1976 році [28]. Вони були колегами в компанії Macbeth. Кілька років тому ця компанія була придбана X-Rite, іншою компанією, що спеціалізується на всьому, що пов'язано з наукою про кольори.

Карта складається з 24 елементів. Спектральні дані кожного елемента доступні з вибіркою при довжині хвилі 1 нм. Також можливо отримати їх координати для різних колірних просторів під різними джерелами світла. У таблиці 1 представлені всі 24 елементи під освітленням D50 [29].

1.1.7 Перетворення хроматичної адаптації (Chromatic Adaptation transform)

Хроматична адаптація - це візуальний механізм, який дозволяє очам адаптуватися до змін спектрального складу світла. Таким чином людські очі здатні зберігати зовнішній вигляд об'єкта при різних умовах освітлення. Типовим прикладом є аркуш паперу. Його зовнішній вигляд залишається таким же при білому, денному світлі, під вольфрамовою лампочкою або флуоресцентному світлі [30].

Таблиця 1 – Перевірка кольору Macbeth при освітленні D50

N.	Color name	Color
1	Dark skin	
2	Light skin	
3	Blue sky	
4	Foliage	
5	Blue flower	
6	Bluish green	
7	Orange	
8	Purplish blue	
9	Moderate red	
10	Purple	
11	Yellow green	
12	Orange yellow	
13	Blue	
14	Green	
15	Red	
16	Yellow	
17	Magenta	
18	Cyan	
19	White 9.5	
20	Neutral 8	
21	Neutral 6.5	
22	Neutral 5	
23	Neutral 3.5	
24	Black	

Для імітації того ж механізму на цифрових пристроях було створено Chromatic Adaptation Transform (CAT). CAT приймає тристимульне значення тестового зразка під випробовуваним джерелом світла і обчислює відповідні значення тристимула під контрольним джерелом світла. У наступних розділах описані два CAT: Перетворення Вон Кріса і CIECAT1994.

Перетворення Вон Кріса

Трансформація Вор Кріса була розроблена Йоханнесом фон Крісом. Цей CAT використовується при розрахунку CRI з використанням «методу пробних зразків», описаного в пункті 4.7.

Формули, які використовуються для конкретного прикладу виглядають наступним чином:

$$u_{c,i} = \frac{10.872 + 0.404\left(\frac{c_r}{c_t}\right)c_{t,i} - 4(d_r/d_t)d_{t,i}}{16.518 + 1.481\left(\frac{c_r}{c_t}\right)c_{t,i} - (d_r/d_t)d_{t,i}} \quad (24)$$

$$v_{c,i} = \frac{5.520}{16.518 + 1.481\left(\frac{c_r}{c_t}\right)c_{t,i} - (d_r/d_t)d_{t,i}} \quad (25)$$

$$c = \frac{(4 - u - 10v)}{v} \quad (26)$$

$$d = \frac{(1.708v - 1.418u + 0.404)}{v} \quad (27)$$

Значення нижніх індексів у формулах:

- r контрольне джерело світла;
- t тестове джерело світла;
- (t, i) досліджуване джерело світла, зразок;
- c поточний досліджуваний зразок кольору.

Перетворення CIECAT94

CIECAT94 – це еволюція перетворення Вон Кріса, що враховує рівень яскравості. Цей CAT використовується для обчислення CRI за допомогою

методу R96a, описаного в пункті 4.8. Тут повідомляється про стандартну процедуру розрахунку цього CAT [30][31]. Нижні індекси означають:

- w біле світло;
- rw контрольне (еталонне) біле світло (джерело);
- c тестовий зразок, обчислений під еталонним білим світлом.

Три головні кроки:

1. Обчислити значення RGB тестового зразка під випробовуваним джерелом світла, використовуючи наступну матрицю:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.40024 & 0.7076 & -0.08081 \\ -0.2263 & 1.16532 & 0.04570 \\ 0 & 0 & 0.91822 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (28)$$

2. Обчислити значення R_c, G_c, B_c під еталонним джерелом світла з використанням наступних формул:

$$R_c = (Y_o \xi_{rw} + n) K^{1/\beta_1(R_{rw})} \left[\frac{R + n}{Y_o \xi' + n} \right]^{\beta_1(R_w)/\beta_1(R_{rw})} - n \quad (29)$$

$$G_c = (Y_o \eta_{rw} + n) K^{1/\beta_1(G_{rw})} \left[\frac{G + n}{Y_o \eta' + n} \right]^{\beta_1(G_w)/\beta_1(G_{rw})} - n \quad (30)$$

$$B_c = (Y_o \zeta_{rw} + n) K^{1/\beta_2(B_{rw})} \left[\frac{B + n}{Y_o \zeta' + n} \right]^{\beta_2(B_w)/\beta_2(B_{rw})} - n \quad (31)$$

3. Обчислити значення тристимула, адаптовані з використанням інверсії матриці, яка використовується для перетворення значень в RGB:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} 1.85995 & -1.12939 & 0.2199 \\ 0 & 0.63881 & 0 \\ 0 & 0 & 1.08906 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (32)$$

Значення ξ_{rw}, η_{rw}, ζ_{rw} та ξ_w, η_w, ζ_w відповідають тестовим та еталонним корелятам хроматичності джерел світла їх хроматичних координат.

$$\xi = (0.48105 x_0 + 0.78841 y_0 - 0.080811)/y_0 \quad (33)$$

$$\eta = (-0.27200x_0 + 1.11962y_0 + 0.04570)/y_0 \quad (34)$$

$$\zeta = 0.91822(1 - x_0 - y_0)/y_0 \quad (35)$$

Значення ξ', η', ζ' є адаптованими значеннями ξ, η, ζ та обчислюються наступним чином:

$$\begin{bmatrix} \xi' \\ \eta' \\ \zeta' \end{bmatrix} = \alpha \begin{bmatrix} \xi_w \\ \eta_w \\ \zeta_w \end{bmatrix} + (1 - \alpha) \begin{bmatrix} \xi_{rw} \\ \eta_{rw} \\ \zeta_{rw} \end{bmatrix} \quad (36)$$

де α є:

$$\alpha = 0.1151 \log_{10} L_a + 0.0025(L^* - 50) + (0.22D + 0.510) \quad (37)$$

У цій формулі L_a це адаптивна яскравість $(\text{кд}/\text{м}^2)$ в тестовій області, а L^* - CIELAB освітленість зразка. Для кольорів об'єкта $D = 1$ і для освітлюваного кольору (luminous color) (зазвичай CRT) $D = 0$. Значення α має бути меншим за одиницю.

R_w, G_w, B_w та R_{rw}, G_{rw}, B_{rw} обчислюються за формулами, вказаними нижче:

$$\begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} = L_a \begin{bmatrix} \xi_w \\ \eta_w \\ \zeta_w \end{bmatrix} \quad (38)$$

$$\begin{bmatrix} R_{rw} \\ G_{rw} \\ B_{rw} \end{bmatrix} = L_{ra} \begin{bmatrix} \xi_{rw} \\ \eta_{rw} \\ \zeta_{rw} \end{bmatrix} \quad (39)$$

L_{ra} встановлено у 63.66 кд/м^2.

β_1 та β_2 функції:

$$\beta_1(I) = \frac{6.469 + 6.362I^{0.4495}}{6.469 + I^{0.4495}} \quad (40)$$

$$\beta_2(I) = 0.7844\left(\frac{8.414 + 8.09I^{0.5128}}{8.414 + I^{0.5128}}\right) \quad (41)$$

Остання, але не менш важлива формула для обчислення K:

$$K = \frac{[(Y_0\xi' + n)/(20\xi' + n)]^{(2/3)\beta_1(R_r)}}{[(Y_0\xi_{rw} + n)/(20\xi_{rw} + n)]^{(2/3)\beta_1(R_{rw})}} \cdot \frac{[(Y_0\eta' + n)/(20\eta' + n)]^{(1/3)\beta_1(G_r)}}{[(Y_0\eta_{rw} + n)/(20\eta_{rw} + n)]^{(1/3)\beta_1(G_{rw})}} \quad (42)$$

Коефіцієнт шуму складає 0.1 у всіх приведених вище формулах.

1.2 Радіометрія

Як видно з попереднього пункту, очі чутливі до невеликої підмножини всього спектра електромагнітного випромінювання з довжиною хвилі від 400 нм до 700 нм – видиме світло. Тому світло являє собою форму електромагнітного випромінювання.

Набір досліджень і методів, які намагаються описати і виміряти спосіб поширення, відображення і передачі електромагнітного випромінювання світла, називається радіометрією. У наступних декількох пунктах представлені деякі основи радіометрії. Це фундаментальна частина комп'ютерної графіки, так як більша частина алгоритму, який використовується в цій області і представлена в цій тезі, спробує оцінити ці величини.

Радіаційний потік іноді просто називають потік, він описує кількість випромінюваної енергії, відбитої або переданої з поверхні в одиницю часу. Енергія випромінювання - це енергія електромагнітного випромінювання. Одиниця виміру потоку дорівнює Джоуль в секунду $Дж/с$ і зазвичай позначається грецькою буквою ϕ.

1.2.1 Інтенсивність випромінювання та здатність світитися (вихід світла)

Іншими двома важливими величинами радіометрії є інтенсивність випромінювання і здатність світитися. Перший описує потік, що надходить на поверхню на одиницю площі. Другий описує потік, що виходить з поверхні на одиницю площі [32].

Формально випромінювання описується наступним рівнянням:

$$E = \frac{d\phi}{dA} \tag{43}$$

де диференційний потік обчислюється над диференціальною областю. Він вимірюється в одиниці ватів на квадратний метр $\text{Вт}/\text{м}^2$.

Свічення є останньою і найбільш важливою величиною радіометрії. Щоб дійсно зрозуміти це визначення, корисно дати визначення тілесного кута (просторового кута – одне і те ж).

Тілесний кут - це розширення двовимірного кута в 3D на одиничній сфері. Це загальна площа, спроектована об'єктом на одиничну сферу, з центром у точці p. Він вимірюється у стерадіанах. Вся одинична сфера відповідає тілесному куту 4π (площа поверхні одиничної сфери). Тілесний кут зазвичай позначається як Ω, але також можна представити його за допомогою ω, тобто безліч всіх векторів у різних напрямках, закріплених в точці p, які вказують на область на одиничній сфері та об'єкті [34]. Тепер можна дати визначення свіченню, тобто щільність потоку на одиницю тілесного кута на одиницю площі:

$$L = \frac{d\phi}{d\omega dA^{\perp}} \tag{44}$$

У цьому випадку $dA^{\perp}$ це спроектована площа dA на поверхню, перпендикулярну до ω. Таким чином свічення описує межу випромінення падаючого світла на поверхню як конус напрямків, які нас цікавлять на поверхню dA, що також стає дуже малою [34].

Корисно провести відмінність між свіченням, що досягає точки, яку часто називають падаючим свіченням і позначається $L_i(p, \omega)$ і свіченням, що залишає точку, звану свіченням вильоту, що позначається $L_0(p, \omega)$. Ця різниця буде використана в рівняннях, описаних в наступних розділах. Важливо також відзначити ще одну корисну властивість, яка пов'язує два типи свічення:

$$L_i(p, \omega) \neq L_0(p, \omega) \tag{45}$$

РОЗДІЛ 2.ТРАСУВАННЯ ПРОМЕНІВ: ФОТОРЕАЛІСТИЧНИЙ РЕНДЕРИНГ

Трасування променів - одна з найпотужніших технологій комп'ютерної графіки. Вона здатна генерувати зображення, які створюють візуальний реалізм, який важко отримати за допомогою стандартного 3D-рендеринга в реальному часі. Як це працює?

Загальний алгоритм трасування променів заснований на тому факті, що в реальному світі світло, що починається від джерела, відбивається між поверхнями об'єктів і надходить в очі людини, які, як раніше бачили, декодують його, щоб отримати колір об'єктів.

Алгоритм трасування променів відслідковує зворотній шлях: починаючи з очей, які зазвичай визначаються як широко використовувана в комп'ютерній графіці камера, алгоритм відстежує промені світла, які відскакують в сцені і надходять до джерела світла, і обчислює колір об'єктів, на яких промені відбивалися, з використанням моделі поверхневого відбиття . Ці моделі будуть описані в розділі 2.3. Таким чином відслідковування всіх різних напрямків світла, прямих і непрямих, трасуючи промені з усіма його варіантами, є глобальною моделлю освітлення.

У наступному пункті буде введена коротка класифікація алгоритмів трасування променів. У TTFD реалізована тільки частина з них: алгоритм Уайтеда і трасування шляху. Але перш ніж починати з класифікації, корисно визначити, як трасування променів (і всі моделі глобального освітлення) намагаються вирішити рівняння рендеринга.

2.1 Рівняння рендеринга

Рівняння рендеринга було введено Джеймсом Каджая в 1986 році [14]. Іноді його також називають LTE – Light Transport Equation. Це рівняння описує рівноважний розподіл свічення в сцені [33].

Рівняння дає повне відбите випромінювання в точці у вигляді суми випущеного і відбитого світла від поверхні. Стандартне формулювання рівняння рендеринга:

$$L_0(p, \omega_0) = L_e(p, \omega_0) + \int_\Omega f_r(p, \omega_i, \omega_0)\, L_i(p, \omega_i) \cos\theta_i \, d\omega_i \qquad (46)$$

де значення кожного компоненту наступне:

- p точка на поверхні у сцені;
- ω_0 напрямок вихідного світла;
- ω_i напрямок вхідного світла;
- $L_0(p, \omega_0)$ свічення вильоту у точці p;
- $L_e(p, \omega_0)$ свічення, що випромінюється у точці p;
- Ω одинична напівсфера, центрована навколо нормалі у точці p;
- $\int_\Omega \ldots d\omega_i$ це інтеграл по одиничній напівсфері;
- $f_r(p, \omega_i, \omega_0)$ це двопроменева функція відбиваючої здатності (Bidirectional Reflectance Distribution Function). Детальний опис цієї функції і її корельованої функції двонапрямленого поверхневого розсіювання відбиття (Bidirectional Transmittance Distribution Function) і різних моделей, які використовуються для її розрахунку, описані в пункті 2.3. Їх реалізація в TTFD буде обговорюватися у розділі 4;
- $L_i(p, \omega_i)$ падаюче випромінювання, що потрапляє в точку p;
- $\cos\theta_i$ задається точковим добутком між ω_i і нормаллю в точці p, і є коефіцієнтом ослаблення випромінення через кут падіння.

Мета всіх глобальних моделей освітлення, а також трасування променів з усіма його варіантами, - вирішити це рівняння.

2.2 Коротка класифікація методів трасування променів

На рисунку 7 наведена класифікація методів.

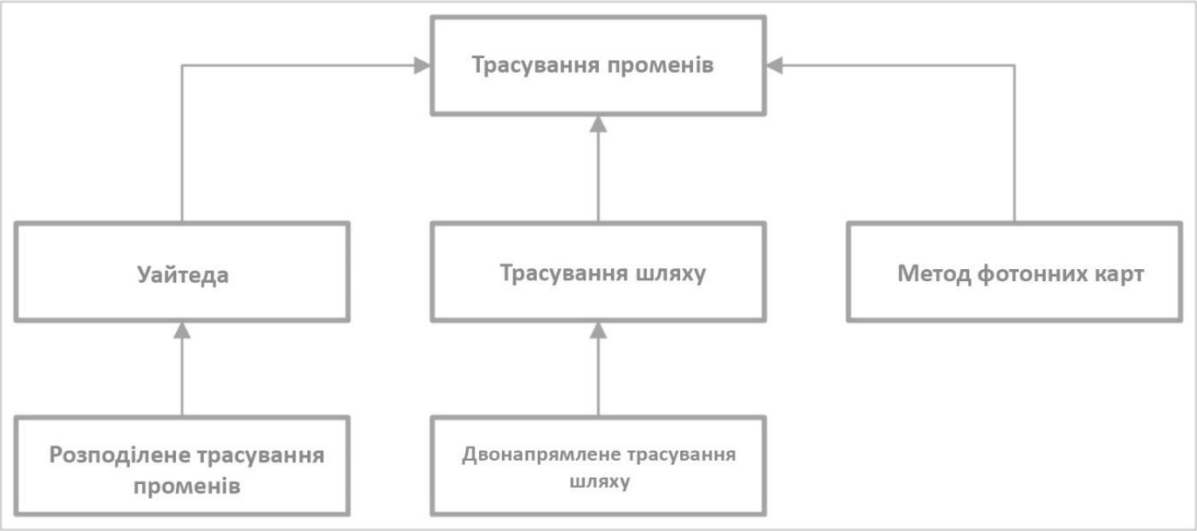

Рисунок 7 – Класифікація методів трасування променів

Перший базовий тип трасування променів зазвичай називають трасуванням Уайтеда. Ця модель була винайдена, як випливає з назви, Тернером Уайтедом в 1979 році [13][14]. Він передбачає, що коли промінь, який починається з камери, перетинає об'єкт, для обчислення компоненти прямого світла використовується модель локального поверхневого відбиття. Але алгоритм йде далі і трасуються три типи променів: промінь відбиття, рефракції і тіньові промінь. Перший тип використовується для світловідбиваючого матеріалу, другий використовується для матеріалу, в якому заломлюються променів, а третій – для розрахунку, якщо точка не досягається безпосередньо світлом, тому вона затінена.

Трасування Уайтеда – не є одним з найреалістичніших методів трасування променів. Інші ефекти і покращення можуть бути додані у результаті відслідковування ще більшої кількості променів. Перш за все – більше тіньових променів можуть бути трасовані до кожної точки в сцені.

Замість того, щоб перевіряти їх пересічення у одній точці на поверхні, що освітлюється, як правило це її центр (або джерело у випадку точкового світла), можна відібрати декілька точок рівномірно і для кожної виконати тест тіньових променів: таким чином можна розрахувати відсоток тіні у точці і створити те, що називають м’якою тінню.

Трасування шляху – це метод з максимальним рівнем візуальної точності з усіх наявних методів трасування променів (рисунок 8). Він був представлений Джеймсом Каджією у 1986 році [14] в якості вирішення рівняння рендеринга, представленого у пункті 2.1. Він використовує метод інтеграції Монте-Карло [33], щоб дати числове рішення цього рівняння.

Більш детально про цей метод буде пояснено у наступних пунктах. Тут буде представлений загальний опис алгоритму.

Для кожного пікселя приймається кілька вибірок. Кожна з цих вибірок обчислюється, трасуючи промінь від камери і слідуючи його відбиттям у сцені, поки промінь не досягне джерела світла. Потім ці вибірки використовуються з методом Монте-Карло для розрахунку кольору кожного пікселя в кінцевому зображенні. Цей метод єдиний, який здатний генерувати безліч візуальних ефектів без необхідності відстежувати вторинні промені. Він також має можливість природним чином моделювати деякі фізичні явища, які зазвичай вимагають певних методів для їх рендеринга. Наприклад, може генеруватися каустика без додаткових обчислень.

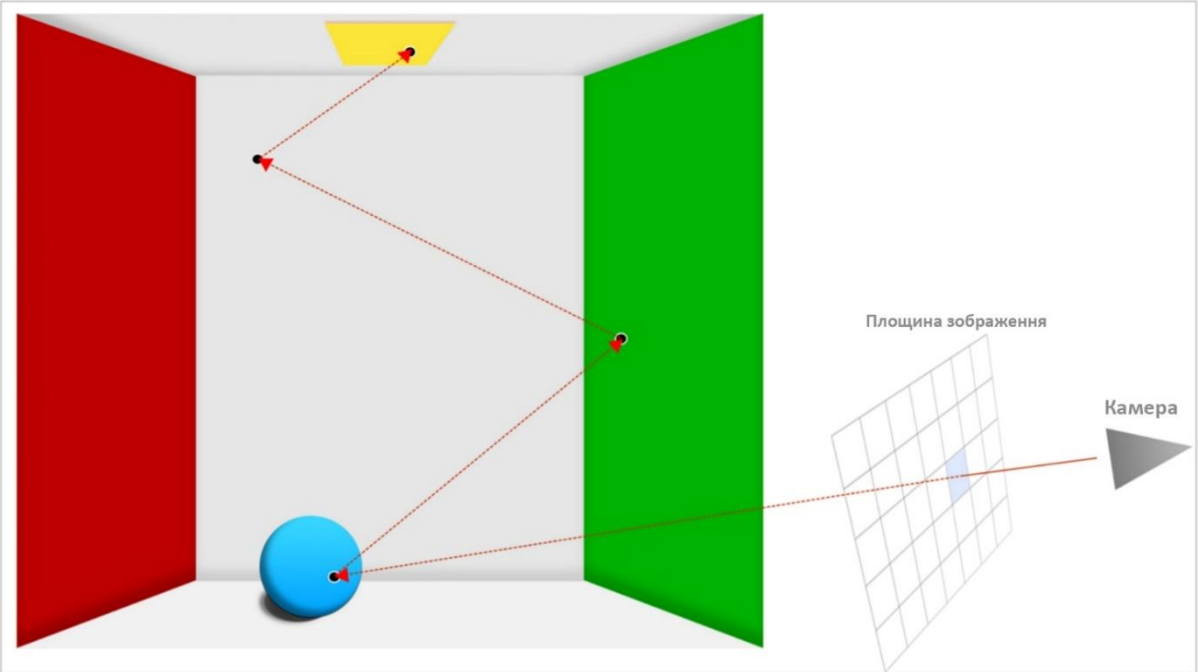

Рисунок 8. Трасування шляху, відслідковування шляху променю від камери до джерела світла

Варіантом трасування шляху є двонапрямлене трасування шляху [34]. У цьому методі промені простежуються в двох напрямках: від камери до джерела світла і від джерела світла до камери. Таким чином сцена може бути візуалізована (пройти процес рендерингу) з високим рівнем візуального реалізму та з більш високою швидкістю, ніж трасування шляху.

2.3 Двопроменева функція відбиваючої здатності (BRDF) та функція двонапрямленого поверхневого розсіювання відбиття (BTDF)

Одним з основних компонентів рівняння рендеринга, описаного у розділі 2.1, є двопроменева функція відбиваючої здатності (BRDF). Ця функція описує відбиття світла від поверхні. Вона представляє собою константу пропорційності між диференційним свіченням вильоту і диференційною освітленістю у точці p. Параметри функції наступні: напрямок падаючого світла, напрямок вихідного світла, точка на поверхні.

Формула функції має вигляд:

$$f_r(p, \omega_i, \omega_0) = \frac{dL_0(p, \omega_0)}{dE(p, \omega_i)} \tag{47}$$

BRDF має дві важливі властивості:

1. Це симетрична функція, звідси для всіх пар напрямків $f_r(p, \omega_i, \omega_0) = f_r(p, \omega_0, \omega_i)$;
2. Вона задовольняє принципу збереження енергії: відбите світло менше чи рівне падаючому світлу.

Деякі специфічні поверхні матеріалів, наприклад скло, відбивають та пропускають світло одночасно. Таким чином, частина світла проходить крізь матеріал. По цій причині існує інша функція – функція двонапрямленого поверхневого розсіювання відбиття (Bidirectional scattering distribution function) визначена тим же чином, що і BRDF (рисунок 9), але з напрямками ω_i та ω_0 розташованими в протилежних

напівсферах навколо p [32]. Зазвичай ця функція позначається як $f_t(p, \omega_i, \omega_0)$.

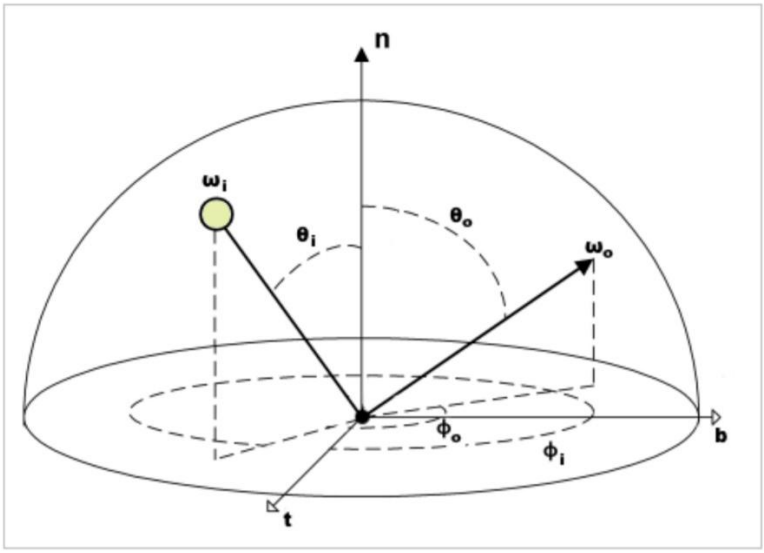

Рисунок 9 – Схема представлення BRDF [35]

Як розраховуються BRDF та BTDF? Був розроблений набір моделей, кожна з яких представляє собою специфічні властивості різних матеріалів.

У цих моделях використовується напрямок падаючого світла ω_i і напрямок вихідного світла ω_0 у сферичних координатах $\omega_i(\theta_i, \phi_i)$, $\omega_0(\theta_0, \phi_0)$ відносно дотичного простору для точки поверхні, де нормалі вирівняні з віссю y. Це відбувається тому, що BRDF визначається у системі координат, локальній для точки на поверхні, що розглядається, тому дотичний простір є очевидним вибором.

Більш детальна інформація про розрахунок сферичних координат міститься в додатку 1.

У наступних розділах показані теоретичні передумови деяких з цих моделей. Це ті, що реалізовані в TTFD. Таким чином, буде легше зрозуміти реалізацію частини його основних функціональних можливостей.

2.3.1 Емпіричні моделі

Перші дві моделі представлені Фонгом і Блінн-Фонгом. Їх називають емпіричними, тому що вони намагаються описати, як поверхня відбиває

світло як наближення спостережуваного явища (approximation of observed phenomenon).

Модель Фонга була запропонована Буі Туонг Фонгом у 1973 [36].

Він описує відбиття світла від поверхні як суму різних компонент, які намагаються пояснити можливу поведінку світла на матеріалі, використовуючи наступні параметри:

1. K_e описує випромінюючу складову матеріалу;
2. K_a описує навколишній (ambient) компонент матеріалу;
3. K_d описує розсіюючу компоненту матеріалу;
4. K_s описує дзеркальну компоненту матеріалу;

Рівняння для обчислення кількості світла у певній точці p на поверхні у сцені з j джерелами світла:

$$I_p = K_e I_{max} + K_a I_{amb} + \sum_{i=1}^{j} K_d I_{i,d}(l_i \cdot n) + K_s I_{i,s}(r_i \cdot v)^n \qquad (48)$$

Вираз $(l_j \cdot n)$ являється затухаючим членом дифузної складової, яка залежить від косинуса кута падіння напрямку світла l_j відносно нормалі поверхні n, що розраховується як добуток точок цих двох векторів.

Вираз $(r_i \cdot v)^n$ контролює дзеркальну компоненту і є добутком точок вектору відбиття r_i і напрямку спостереження, описаного вектором v. Перший розраховується за наступною формулою:

$$r_i = 2(n \cdot l_i)n - l_i \qquad (49)$$

Вектор v розраховується як різниця між точкою зору камери і точкою p. Показник члена дзеркального контролю використовується для управління розміром дзеркальної підсвітки.

Ця модель зазвичай використовується у трасуванні променів Уайтеда для рендеринга сцени визначеної RGB значеннями. По цій причині у TTFD, як буде описано пізніше, модель Фонга буде використовуватися для дослідження рендеринга зображення зі стандартними RGB кольорами.

Модель Блінна-Фонга це еволюція моделі Фонга, запропонована Джеймсом Блінном у 1977 [37].

Зокрема, дзеркальна складова апроксимується з допомогою кута γ між нормаллю і половинним вектором h вектору напрямку спостерігача v і вектором напрямку світла l. Отже h розраховується за наступною формулою:

$$h = \frac{l + v}{|l + v|} \tag{50}$$

а дзеркальна компонента розраховується як:

$$I_s = K_s I_s (h \cdot v)^n \tag{51}$$

Як і для моделі Фонга, модель Блінна-Фонга буде використана у зв'язці з трасуванням променів Уайтеда для сцени зі стандартними RGB кольорами.

2.3.2 Фізичні моделі (Physically based models)

Фізичні моделі BRDF засновані на законах фізики, пов'язаних зі взаємодією світла з різними поверхнями. У наступних розділах представлено опис серії фізично обґрунтованих (PB) моделей. Вони можуть використовуватися для нанесення широкого спектру матеріалів, в тому числі на скляні, пластикові і дифузні поверхні. Всі ці моделі реалізовані в TTFD.

І останнє, але не менш важливе: моделі, представлені в цій роботі, ізотропні: властивості відбивної здатності поверхні не змінюються відносно обертання поверхні навколо нормалі поверхні. Моделі, які враховують бажаний дотичний напрямок, називаються анізотропними. TTFD не підтримує анізотропну модель.

Модель Ламберта була першою фізично обґрунтованою BRDF. Вона моделює ідеальну розсіювальну поверхню, що відбиває падаюче світло однаково у всіх напрямках. Хоч це і не цілком фізично правдоподібно (насправді це іноді включається у емпіричні моделі), це гарне наближення до багатьох справжній поверхонь.

Враховуючи SPD об’єкта, оголошеного тут як R, стандартна формула для розрахунку моделі Ламберта наступна:

$$f_r(p, \omega_i, \omega_0) = \frac{R}{\pi} \tag{52}$$

Число π використовується для підтвердження того, що модель Ламберта слідує принципу збереження енергії: у цьому випадку вихідне світло, що розподілене по всій напівсфері, не перевищує падаючого світла.

Модель Орена-Найара використовується для представлення розсіювальних поверхонь. Він був визначений Майклом Ореном та Шрі К. Найара в 1994 році [9]. Вона є частиною великого набору моделей під назвою «Моделі Microfacets», оскільки вони описують поверхні, що складаються з колекції невеликих мікрограней з певними геометричними властивостями.

Ця модель більш реалістична, ніж модель Ламберта. Це пояснює той факт, що грубі поверхні виглядають яскравіше, коли напрямок світла наближається до напрямку спостереження. Він описує поверхню, складену з колекції V-образних ламбертівських граней, розподілених за розподілом Гаусса з параметром σ, який описує стандартне відхилення від кута орієнтації [38].

Модель враховує ефекти локального освітлення між мікрогранями, такі як затінення, взаємне відбиття і маскування, і описується наступною формулою (де R це SPD об’єкта):

$$f_r(p, \omega_i, \omega_0) = \frac{R}{\pi}(A + Bmax(0, \cos(\phi_i - \phi_0)) \sin\alpha \tan\beta) \tag{53}$$

Враховуючи σ у радіанах, різні члени попереднього рівняння визначаються як:

$$A = 1 - \frac{\sigma^2}{2(\sigma^2 + 0.33)} \tag{54}$$

$$B = \frac{0.45\sigma^2}{\sigma^2 + 0.09} \tag{55}$$

$$\alpha = max(\theta_i, \theta_0) \quad (56)$$

$$\beta = min(\theta_i, \theta_0) \quad (57)$$

Модель дзеркального відбиття BRDF і **модель дзеркального пропускання BTDF** описують гладку поверхню, яка, як випливає з назви, породжує явище відбиття і заломлення світла. Для напрямку падаючого променю ω_i світло відбивається або переломлюється в одному напрямку ω_0, який повинен бути розрахований і використаний в якості нового променя для трасування у всіх описаних раніше алгоритмах трасування променів. Для відбиття цей напрямок задається рівнянням 49, яке використовується в моделях Фонга і Блінна-Фонга. Для пропускання вихідний напрямок задається законом Снелла, який пов'язує кут між нормаллю і напрямом пропускання θ_t, і кутом між нормаллю і напрямком падаючого світла θ_i:

$$\eta_i \sin\theta_i = \eta_t \sin\theta_t \quad (58)$$

У приведеній формулі η це індекс заломлення матеріалу і описує наскільки повільніше світло проходить через матеріал у порівнянні з вакуумом [38]. Формула для розрахунку напрямку пропускання:

$$t = \omega_i \frac{\eta_i}{\eta_t} + \left(\frac{\eta_i}{\eta_t} \cos\theta_i - \sqrt{1 - \left(\frac{\eta_i}{\eta_t}\right)^2 (1 - \cos^2\theta_i)} \right) n \quad (59)$$

Зазвичай матеріали ніколи не являються повністю відбиваючими чи пропускаючими. Вони описуються одночасно з кожною з моделей в цьому розділі. Як розраховується розподіл між відбитим і переломленим світлом? Відповіддю є рівняння Френеля. Вони описують відсоток відбитого світла F_r від поверхні. Існує два типи рівнянь Френеля, одні для діелектриків, тобто матеріалів, що не проводять струм, і провідників. TTFD реалізує тільки діелектрики, зокрема скло, тому тут описується рівняння для цих наборів матеріалів. Формула для діелектричного відбиття Френеля:

$$F_r = \frac{(r_\parallel^2 + r_\perp^2)}{2} \tag{60}$$

де члени $r_\parallel^2$ та $r_\perp^2$ розраховуються за наступними рівняннями:

$$r_\parallel = \frac{\eta_t \cos\theta_i - \eta_i \cos\theta_t}{\eta_t \cos\theta_i + \eta_i \cos\theta_t} \tag{61}$$

$$r_\perp = \frac{\eta_i \cos\theta_i - \eta_t \cos\theta_t}{\eta_i \cos\theta_i + \eta_t \cos\theta_t} \tag{62}$$

Як наслідок принципу збереження енергії світло, яке передається з поверхні матеріалу:

$$F_t = 1 - F_r \tag{63}$$

Нагадуємо, що кожна з двох моделей описується певним вихідним напрямком. Це приводить до того, що BRDF та BTDF дорівнюють 0 кругом, крім дзеркального відбиття чи заломлення. Яку функцію можна використати для опису такої поведінки? Дельта функція Дірака. Це функція, яка дорівнює нулю всюди, крім 0 з інтегралом від одиниці по всій дійсній лінії (with an integral of one over the entire real line) [39]. Важливою властивістю цієї функції є те, що [38]:

$$\int f(x)\delta(x - x_0)\,dx = f(x_0) \tag{64}$$

Використовуючи цю властивість, можна вирішити рівняння рендеринга і знайти значення BRDF і BTDF, використовуючи дельта-функцію Дірака, центровану на векторі напрямке дзеркального відбиття або заломлення.

Можливо дати визначення рівнянням цих двох моделей.

Для дзеркального відбиття BRDF описана наступна формула:

$$f_r(p, \omega_i, \omega_0) = RF_r \frac{\delta(\omega_i - r)}{|\cos\theta_i|} \tag{65}$$

де r - вектор дзеркального відбиття для ω_0, описаний у рівнянні 49. Додатковий член косинус у знаменнику необхідний для того, щоб компенсувати той, що взятий з рівняння рендеринга. Для дзеркального

пропускання BTDF описується наступною формулою, де t – вектор пропускання для ω_0, описаний у рівнянні 59:

$$f_r(p, \omega_i, \omega_0) = R\left(\frac{\eta_i}{\eta_t}\right)^2 (1 - F_r)\frac{\delta(\omega_i - t)}{|\cos\theta_i|} \quad (66)$$

Модель Торранса-Спарроу була розроблена К. Е. Торрансом та Е. М. Спарроу у 1967 [10]. Вона належить до набору «моделі мікрограней», як і Орена-Найара. Вона моделює поверхню як набір ідеально гладких мікрограней. Рівняння, що описує цю модель має наступний вигляд:

$$f_r(p, \omega_i, \omega_0) = R\frac{D(\omega_h)G(\omega_0, \omega_i)F_r}{4\cos\theta_0\cos\theta_i} \quad (67)$$

Член $D(\omega_h)$ це функція розподілу, яка описує ймовірність того, що мікрогрань має певну орієнтація відносно половини вектора ω_h. Зокрема, ця модель використовує розподіл Блінна. Цей розподіл використовується в емпіричній моделі, раніше представленої Блінном-Фонгом, запропонованій Джеймсом Блінном у 1977 році [37]. Щоб забезпечити його фізичну правильність, розподіл нормалізується так, що інтеграл по проекційній площі всіх мікрограней дорівнює 1. Його рівняння має наступний вигляд:

$$D(\omega_h) = \frac{e + 2}{2\pi}(\omega_h \cdot n)^e \quad (68)$$

Член $G(\omega_0, \omega_i)$ характеризує геометричне затухання, який передбачає, що мікрограні розподілені вздовж v-подібних канавок. Рівняння, яке використовують для його опису:

$$G(\omega_0, \omega_i) = \min\left(1, \min\left(\frac{2(n \cdot \omega_h)(n \cdot \omega_0)}{\omega_0 \cdot \omega_h}, \frac{2(n \cdot \omega_h)(n \cdot \omega_i)}{\omega_0 \cdot \omega_h}\right)\right) \quad (69)$$

F_r це член рівняння Френеля, той же, що і для діалектрика, раніше представленого для моделі дзеркального відбиття та пропускання.

BRDF також може бути **виміряна**, а отримані дані можуть бути використані для рендеринга матеріалів з різними властивостями. Пристрій, який використовується для вимірювання BRDF являє собою геніорефлектометр (рисунок 10, 11). Різні формати даних BRDF доступні в

мережі Інтернет. Наприклад, є база даних MERL з більш ніж сотнею BRDF даних, готових до використання.

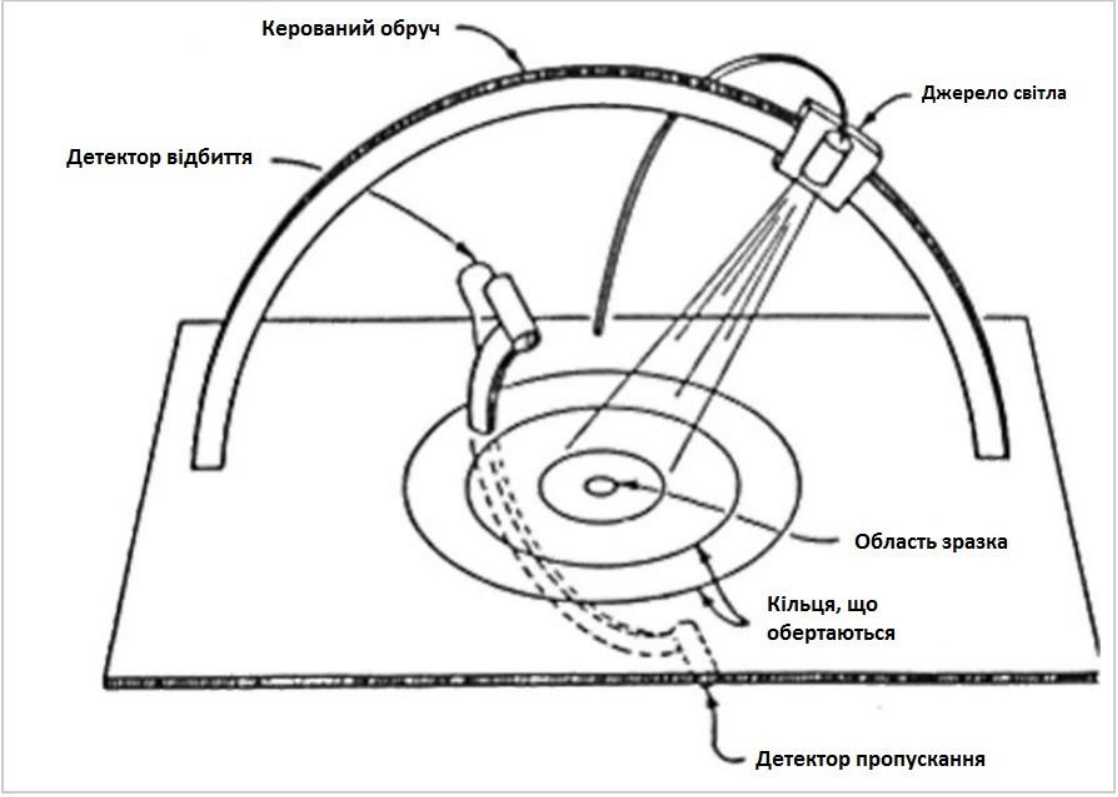

Рисунок 10 – Принцип роботи геніорефлектометра

Рисунок 11 – Геніорефлектометр

Ця база даних зберігає інформацію тільки у RGB форматі. TTFD використовує виміряну BRDF тільки у зв’язці зі спектральними даними для врахування максимального рівня точності кольору.

Ось чому виміряний набір BRDF даних обраний Корнелльським університетом в якості даних відбиття [12].

Цей тип нерегулярних спектральних даних, відібраних на 10 нм (sampled at 10nm), індексується з падаючими і вихідними напрямками світла, записаними як сферичні координати. Ці дані можуть використовуватися з простим алгоритмом інтерполяції найближчих вибірок або з конкретним відображенням даних (data mapping).

Конкретне співставлення (mapping) може бути корисним для більш точного вибору вибірок спектра з виміряних даних для кожного напрямку. В результаті можна уникнути генерації артефактів для поверхні, що рендериться. Цей спосіб використовує TTFD. Докладніше буде описано у наступних розділах.

РОЗДІЛ 3. TTFD: АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ

Всі описані вище поняття, визначення і рівняння складають основу для механізму трасування променів, розробленого для дипломного проекту: TTFD, Test Tracer for Diploma.

TTFD підтримує класичні RGB методи рендеринга з емпіричними BRDF моделями, але його ключові функції наступні:

- Фізичний рендеринг, використання фізичних моделей BRDF і спектральних даних;
- Розрахунок CRI для джерела світла, яке використовується у сценах PBR, які відрендерилися зі спектральними даними.

Таким чином TTFD здатний рендерити різні сцени з нав'язливим фокусом на точність кольору і візуальний реалізм. У наступних пунктах будуть описані всі його функції, зосереджені на деталях реалізації. Перш за все, загальний огляд архітектури TTFD буде показаний у наступному пункті.

3.1 Архітектура ядра

Як вже говорилося раніше, TTFD підтримує кілька видів рендеринга на основі різних типів даних кольору, стандартних RGB чи спектральних даних. Тому необхідно, щоб TTFD міг змінити метод, який використовується для рендеринга сцени зручним способом. Це також причина того, що TTFD повинен дотримуватися принципу розділення проблем (принцип проектування), який потребує розбиття програмного забезпечення на модулі, кожний з яких враховує певний набір інформації і операцій [40]. TTFD складається з різних модулів, кожний з яких створений для управління певною частиною рендерингового рушія трасування променів.

Головний клас, який запускає процес рендеринга – це клас TTFD. Він містить основний цикл загального алгоритму трасування променів. Цей цикл проходить через кожний піксель площини зображення (огляду, view area) і відслідковує поточний промінь камери з екземпляром *Tracer* для

отримання кольору пікселя (або частини пікселя, у випадку рендеринга зі згладжуванням) зображення з допомогою метода *getColor (Ray ray, int bounce)*, який має два параметри: перший – промінь, який повинен бути відстежений, реалізований у класі *Ray* і описується початком і напрямком, другий – максимальна кількість відбиттів, дозволених променю, і після кожного відбиття зменшується. Ці відбиття можуть бути згенеровані по різних причинах: відбиття променя між відбиваючими/заломлюючими матеріалами у моделі трасування променів Уайтеда або для загальної максимальної кількості відбиттів, дозволених при трасуванні. Очевидно, що у класі TTFD існує також загальне налаштування, необхідне для рендеринга сцени: визначення площі зображення (клас *ViewPlane*) і визначення камери (клас *Camera*).

У TTFD реалізовані декілька видів трасувальників, кожний з яких враховує різні особливості рушія. Існує базовий клас Tracer (рисунок 12), який реалізує метод, загальний для всієї спеціалізації трасувальника: *closestIntersection(Ray ray)*. Цей метод використовується для розрахунку найближчого пересічення променя з об’єктами у сцені. Найближчий об’єкт буде відображатися.

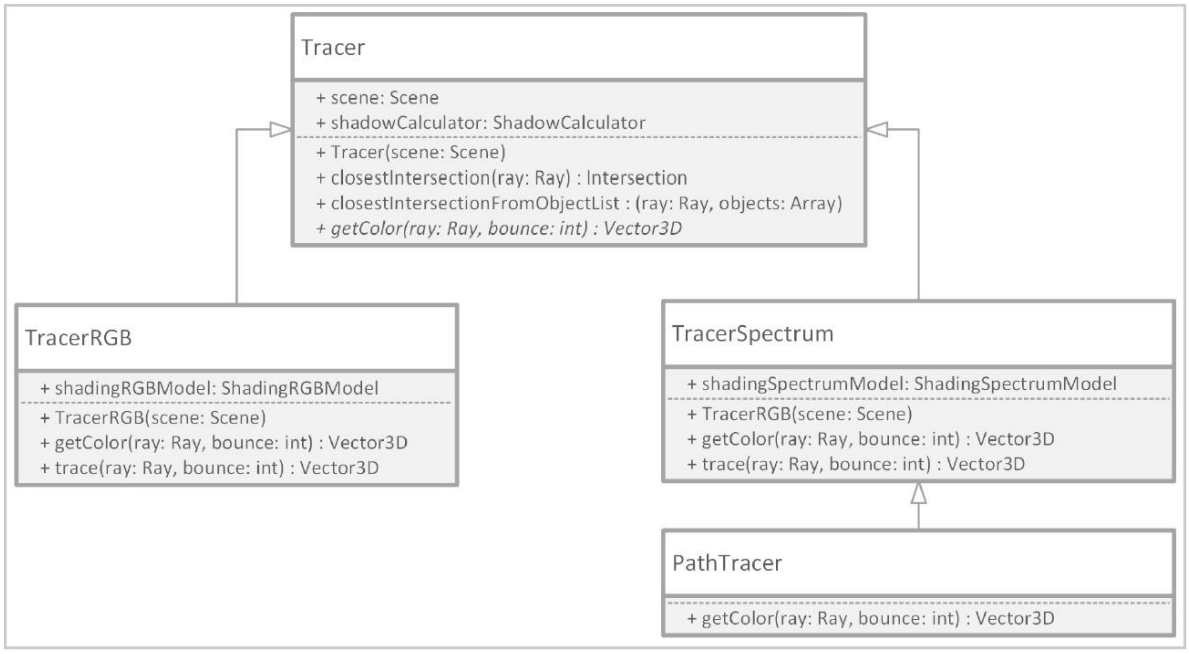

Рисунок 12 – Ієрархія класів Tracer. UML діаграма класів

Це стандартний метод, загальний для всієї моделі трасування променів, реалізований у TTFD. Саме тому він знаходиться у базовому класі.

Існують дві спеціалізації класу *Tracer*, кожна з який враховує різноманітну техніку передачі кольору: *TracerRGB* та *TracerSpectrum*. Як видно з назви, перший використовується для рендеринга сцени з використанням стандартних даних RGB, другий – з використанням спектральних даних. Зокрема, клас *TracerSpectrum* використовує клас *CIE1931XYZ* для розрахунку значень тристимула для спектральних сцен PBR.

Підкласи *Tracer* реалізують два методи: раніше описаний *getColor(Ray ray, int bounce)* і метод *trace(Ray ray, int bounce)*. Останній – це метод, котрий власне виконує трасування променя.

Спеціалізовані класи *Tracer* мають властивість *shadingModel* (рисунок 13), яка зберігає поточну модель затінення, яка буде використовуватися трасувальником при розрахунку кольору.

Для моделей затінення є базовий клас *ShadingModel*. Цей клас розширений двома спеціалізаціями: *ShadingRGBModel* та *ShadingSpectrumModel*. Всі моделі затінення, реалізовані у TTFD, повинні розширити один з цих класів, щоб оголосити підтримку RGB або спектральних даних.

TTFD підтримує наступні моделі затінення, кожна з яких реалізована у своєму класі:

- *Whitted*, клас, який реалізує трасування променій Уайтеда, використовуючи емпіричну BRDF модель Фонга/Блінна-Фонга і дані RGB;
- *WhittedBRDF*, клас, який реалізує трасування променів Уайтеда, використовуючи фізичні BRDF і спектральні дані;

- *PathBRDF*, клас, який реалізує модель трасування шляху, як і для *WhittedBRDF*, використовує фізичні BRDF і спектральні дані для рендеринга сцени.

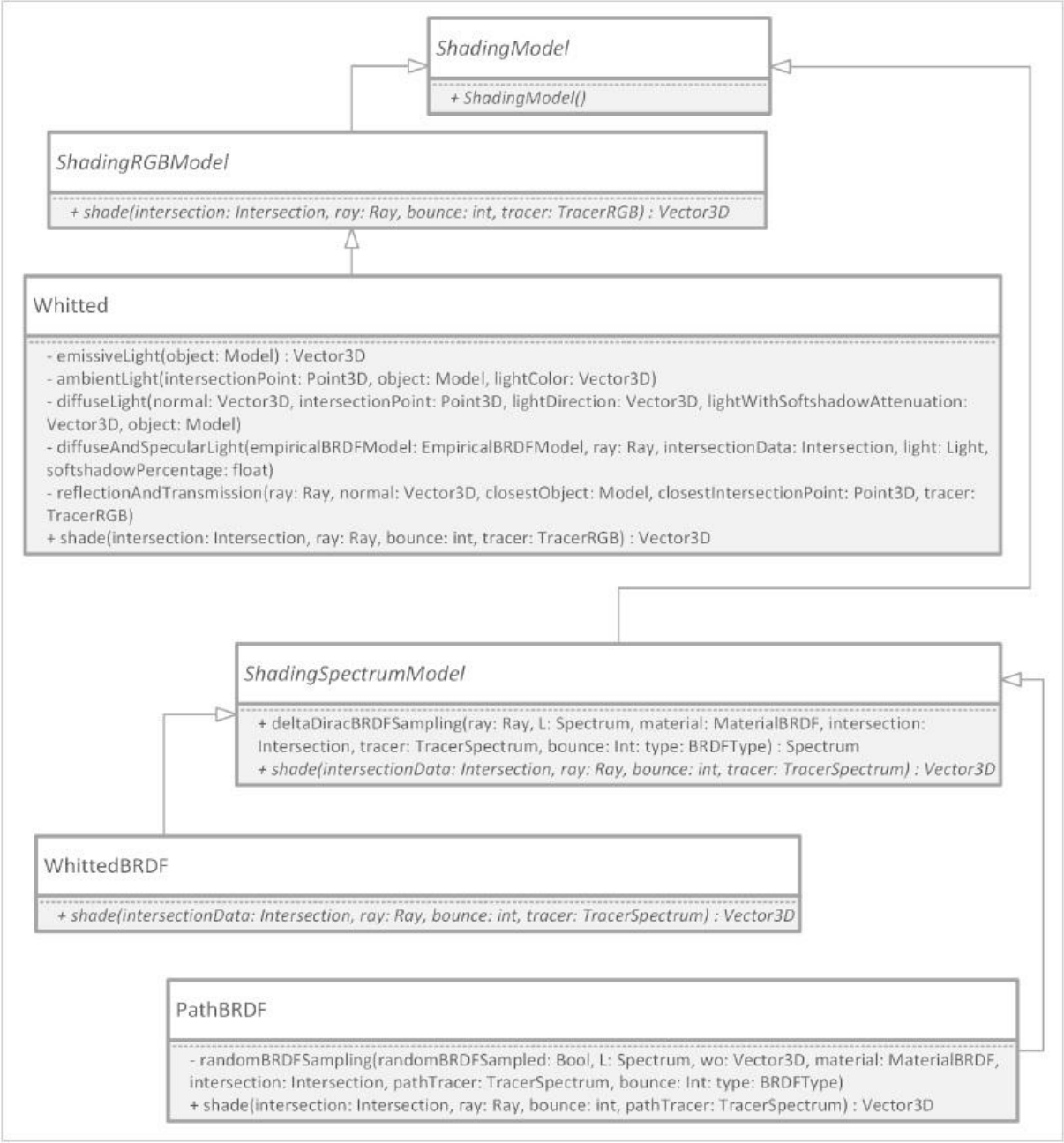

Рисунок 13 – Ієрархія класів ShadingModel. UML діаграма класів

Для підтримки деяких з цих моделей може знадобитися спеціалізований клас трасувальника. Таким чином можна підкласифікувати класи *TracerRGB* та *TracerSpectrum* для налаштування частини процесу трасування або розрахунку кольору. Наприклад, для підтримки трасування шляху, спеціалізації класу *TracerSpectrum*, був

створений клас *PathTracer* для налаштування розрахунку кольору і процесу трасування для цієї моделі.

Всі емпіричні BRDF моделі включені до складу класу *EmpiricalModel:* це вибір був зроблений тому, що головною увагою TTFD є спектральний рендеринг, а рендеринг RGB був доданий як доказ концепції для того, щоб зробити порівняння з PBR сценою.

Фізичні моделі реалізовані як спеціалізація базового класу *BRDF*. Що стосується компонент затінення і трасувальника, додавання нової BRDF моделі потребує тільки створення нового класу, який розширює раніше визначений базовий клас *BRDF* (рисунок 14). TTFD підтримує по замовчуванню наступні фізичні моделі BRDF:

- Ламберта;
- Орена Найара;
- Дзеркального відбиття;
- Дзеркального пропускання;
- Торренса Спарроу;
- Виміряного BRDF.

Ці моделі були описані у розділі 2.3.2. Оскільки основною задачею TTFD являється створення спектральної сцени, всі моделі BRDF реалізовані з підтримкою тільки спектральних даних.

Різні BRDF поєднані у класі *MaterialBRDF* для створення різноманітних типів матеріалів. Також є клас *MaterialRGB,* який використовується у RGB-сцені. Всі вони є спеціалізацією класу *Material*.

Всі попередні компоненти використовують дані, взяті з класу *Scene*, який містить певні сцени з інформацією про модель трасування і затінення, яка буде використовуватися під час рендеринга. Класи *TracerModelFactory* та *ShadingModelFactory* створюють екземпляр класу *Scene*, створений у класі *TTFD*, на основі вибору користувача, який передається як вказівник на всі класи, які він потребує.

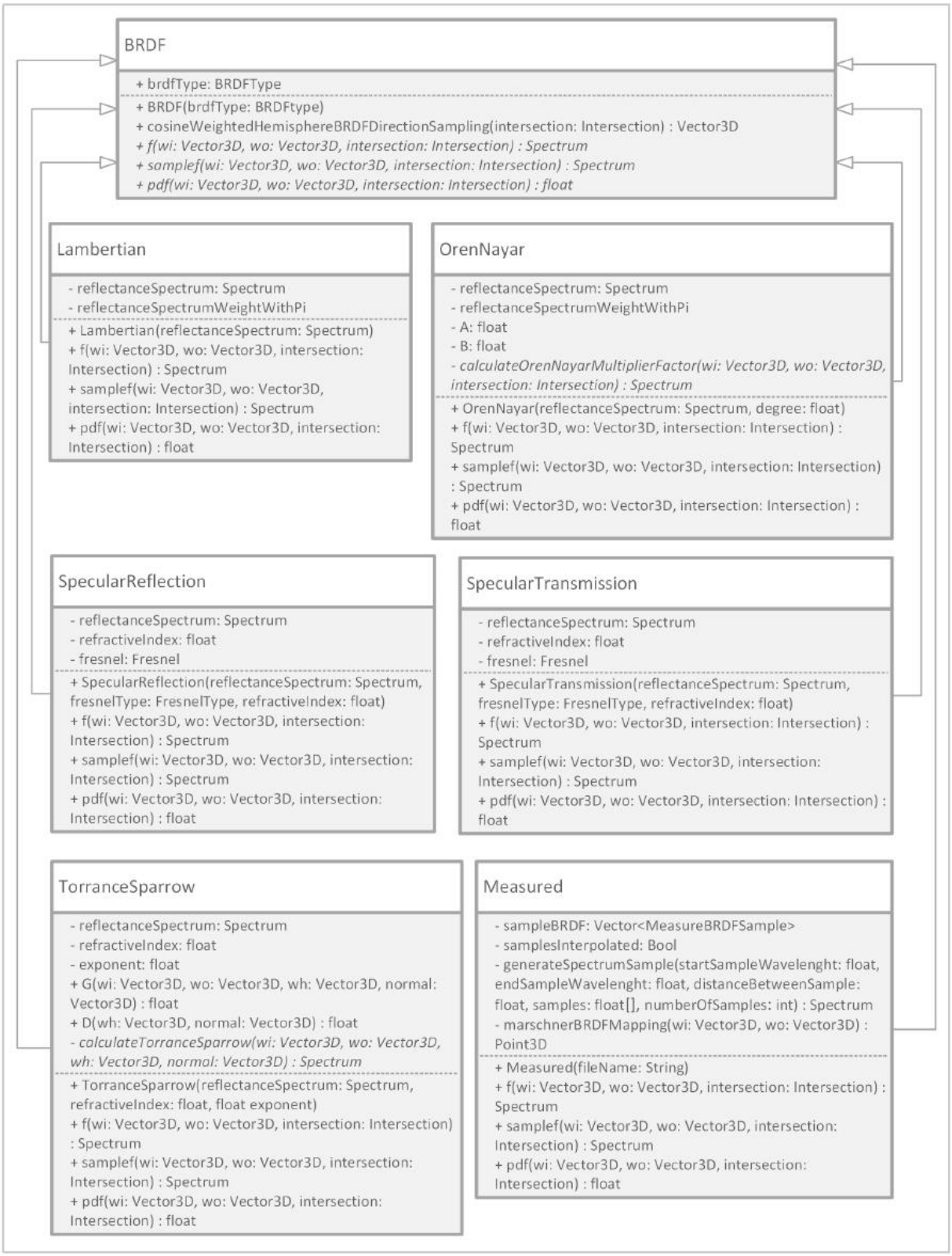

Рисунок 14 – Ієрархія класів BRDF. UML діаграма класів

І останнє, але не менш важливе: для розрахунку CRI сцени використовується клас *ColorRenderingIndex*. Він використовує набір класів, які реалізують всі кольорові простори, представлені у розділі 1, для

виконання розрахунків. Одним з них є раніше визначений клас *CIE1931XYZ*. Він також використовує клас *ChromaticAdaptationTransform*, який реалізує різноманітні перетворення хроматичної адаптації, описані раніше. Існує також набір класів, який просто містить майже всі спектральні дані, які використовуються в TTFD. Ці спектральні дані використовуються при розрахунку CRI і для визначення SPD матеріалів і джерел світла. Нарешті, клас *ColorMatchingFunction,* який використовується класом *CIE1931XYZ,* визначив функцію узгодження кольорів, яка використовується у TTFD.

В цілому, вся представлена структура буле визначена з урахування того, що TTFD буде «підключи і розширюй» системою (pluggable and extensible system): було б легко додати нові трасування і моделі затінення в якості спеціалізованих базових класів, описаних раніше.

Інші компоненти TTFD не перераховані у цьому параграфі, оскільки їх реалізація проста. Наприклад, SPD джерел світла і об’єктів представлена класом *Spectrum*, який являється всього лиш обгорткою для масиву, який містить значення для кожної довжини хвилі.

3.2 Сучасна крос-платформна розробка

У TTFD є частина, не пов’язана строго з комп’ютерною графікою, котра відрізняє її від інших механізмів трасування променів. Зазвичай результат рендеринга в стандартному рушії трасування променів зберігається на диску як зображення, а процес рендеринга запускається і керується з командного рядка. Таким чином, більша частина рушіїв трасування променів, доступних в режимі онлайн є тільки додатком до командного рядка. Але це не відноситься до TTFD, який був створений як додаток для командного рядка операційної системи Linux, але також повноцінний програмний додаток, який може працювати з певним вбудованим інтерфейсом на пристроях Apple, на мобільній операційній системі iOS (рисунок 15) і на робочому столі системи OS X (рисунок 16), а також на платформі Windows.

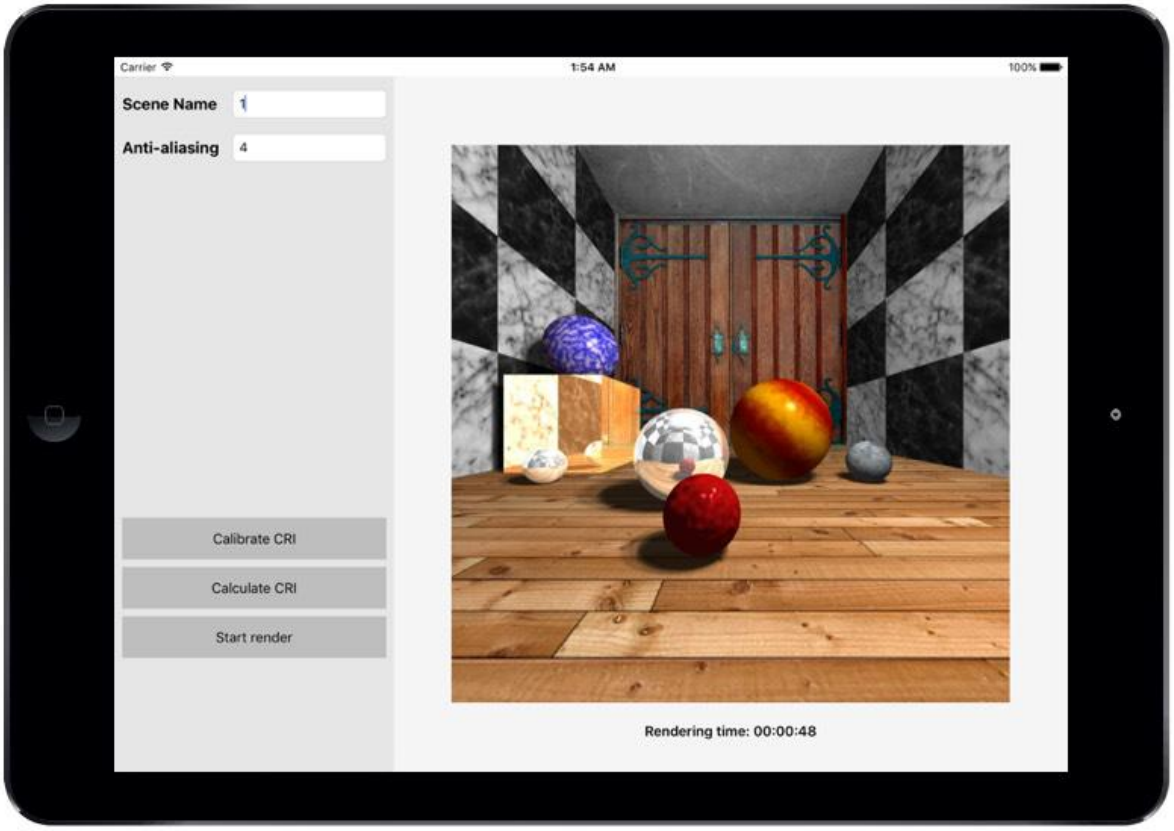

Рисунок 15 – Користувацький інтерфейс в iOS

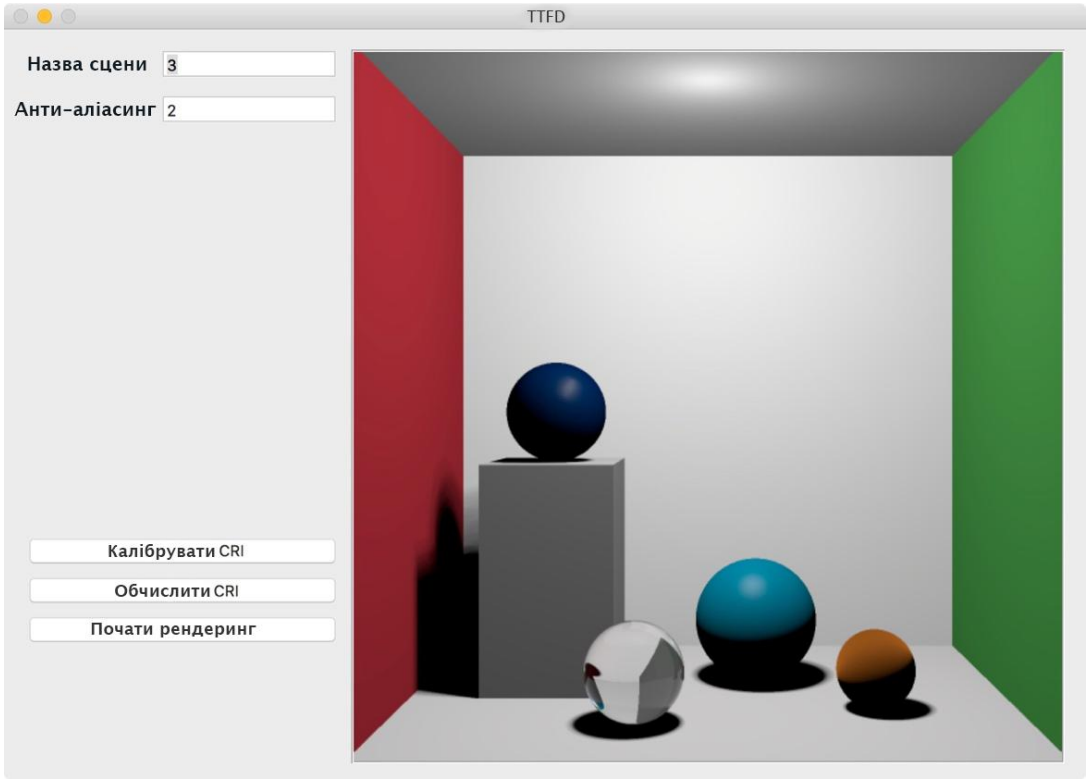

Рисунок 16 – Користувацький інтерфейс у OS X

Реалізація основних частин ядра TTFD були виконана з використанням мови програмування C++. Використовуючи тільки функції, доступні в стандартній бібліотеці C++, можна скомпілювати основні класи з різними компіляторами: g++ зі збірника компіляторів GNU (GCC) на платформах Linux, Visual C++ на платформах Microsoft і LLVM на платформах Apple. Введення сцени в кожному застосунку виконується з використанням конкретного XML-файлу. Додаткову інформацію про визначення сцени TTFD дивитись у додатку В. У наступних параграфах повідомляється короткий опис конкретних реалізацій на кожні платформі.

3.2.1 Apple: iOS і OS X

Головними мовами для розробки власних додатків на платформах Apple є Objective-C та Swift.

Перша – це надбудова мови C з функціями обміну повідомленнями типу Smalltalk. Друга була додана нещодавно як вибір розробника компанією Apple (мова була вперше представлена для розробників у 2014 році). Swift була представлена як потужна мова з набором функції, успадкованих від: C#, Ruby, Rust і знову Objective-C.

Отже, зрозуміло, що перед початком розробки було велике питання: яка з двох мов є правильним вибором? Відповідь проста: Objective-C. Навіть якщо стане зрозуміло, що Apple будуть приділяти більше уваги Swift у майбутньому, оскільки вона намагається привабити більше розробників (тому що це «повільніша» мова), Objective-C буде підтримуватися дуже тривалий період, так, як деякі конкретні додатки завжди будуть потребувати деяку частину функцій, такі, як легка інтеграція з мовами C/C++. Зокрема, для C++ розширення Objective-C, Objective-C++ дозволяє вільно змішувати дві мови з дуже невеликими обмеженнями.

Таким чином, повна архітектура TTFD на пристроях Apple складається з трьох різних рівнів:

1. TTFD Engine Core, який включає класи ядра і особливості рушія трасування променів;
2. Rasterizer клас, який виконує пастеризацію даних зображення від TTFD ядра;
3. Інтерфейс користувача (User Interface) і відображення зображення.

Останнє питання пов’язане з пристроями, що підтримуються. Зрозуміло, що програмний додаток, такий як TTFD, набагато зручніше використовувати на пристрої з достатнім розміром екрану і розширенням. Саме тому TTFD підтримує будь-який комп’ютер Apple під управлінням OS X EI Capitan 10.11 (рисунок 15) чи більш пізньої версії і будь-яку модель iPad, яка може працювати під управлінням iOS 9.0 (рисунок 16) або вище. На даний момент TTFD не підтримує iPhone.

3.2.2 Windows

З релізом Windows 10 багато чого змінилося у світі розробки Microsoft. Фактично, основний випуск їхньої операційної системи привносить нову повністю перероблену платформу для розробки: Універсальні Windows додатки. А це значить, що тепер стало можливим розробляти один програмний додаток, який може бути встановлений на мобільних телефонах, планшетах і настільних комп’ютерах під управлінням Windows 10. Існують дві основні мови програмування, які можна використовувати для розробки UWP (universal Windows platform) додатків: C# та C++. Перша – це «головна» мова Microsoft. Вона була створена як конкурент для Java і стала базовою мовою розробки на платформі Windows. Друга - це мова посилань на платформі Windows впродовж багатьох років. Вона була перевершена C#, але її присутність залишається сильною і на ній все ще написано величезну кількість програмний додатків. В цьому випадку вибір також простий: C++. Таким чином C++ ядро TTFD можна легко інтегрувати у додаток Windows UWP.

Архітектура застосунку TTFD у UWP схожа на ту, що раніше розглядалася для пристроїв Apple, з трьома головними компонентами:

TTFD Core Engine, Rasterizer, специфічний для платформи Windows і, нарешті, інтерфейс користувача і зображення (рисунок 17). Відносно останнього компонента важливо відмітити: користувацький інтерфейс для застосунку UWP був розроблений з використанням XAML, мови на основі XML, який використовується на платформі Microsoft для розробки користувацького інтерфейсу для комп'ютерів та планшетних пристроїв. Це можливо тому, що XAML дозволяє розробнику створити адаптивний макет, який добре адаптується до пристрою, на якому він відображається.

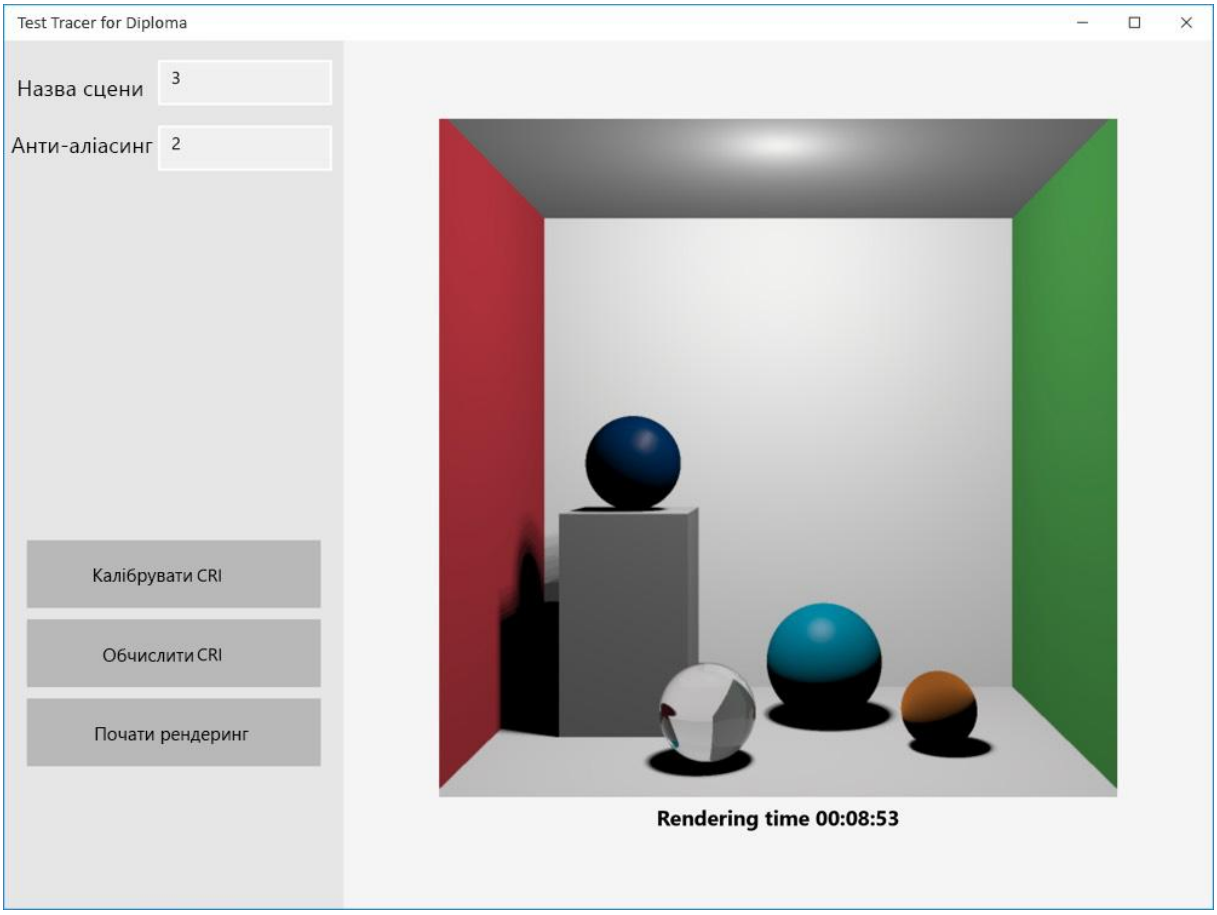

Рисунок 17 – Користувацький інтерфейс у Windows

3.2.3 Linux

Linux – найбільш використовувана операційна система для наукових досліджень. Як було сказано раніше, TTFD також є навчальним інструментом. По цій причині TTFD також доступний в якості додатку для командного рядка, щоб студенти і академічні дослідники використовували його у своєму дистрибутиві.

У зв’язку з тим, що Linux також є найбільш використовуваною операційною системою на серверах, ця версія TTFD може використовуватися для створення найбільш складних сцен з використанням куди більш потужних машин у порівнянні з класичними настільними/переносними комп’ютерами.

Компонент пастеризатора в даному випадку є сторонньою бібліотекою: LodePNG [41]. Це хороший вибір, тому що таким чином головний TTFD engine можна використовувати як автономний додаток, який не залежить від якої-небудь функції операційної системи.

3.3 Архітектура TTFD

У наступні діаграмі показана повна архітектура TTFD (рисунок 18).

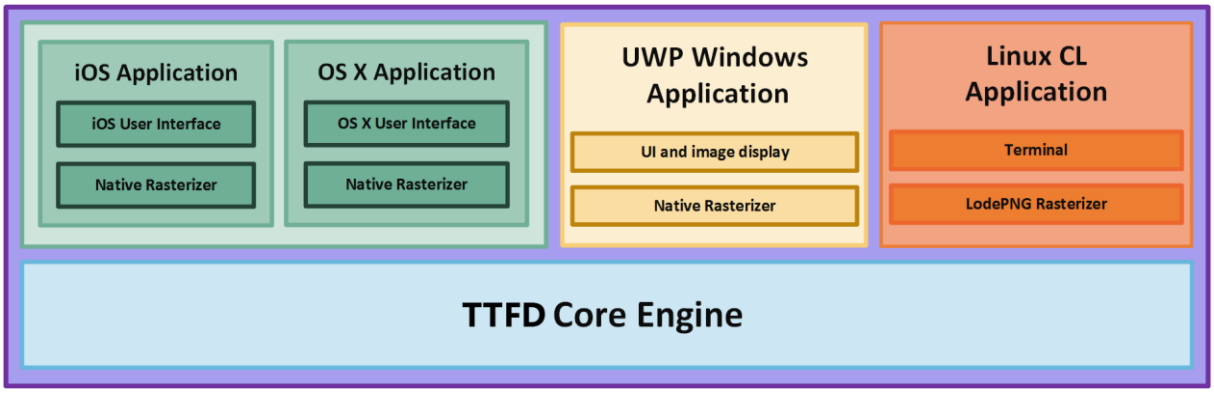

Рисунок 18 – Архітектура продукту

TTFD є крос-платформний додатком. Він був розроблений без використання будь-яких зовнішніх крос-платформних фреймворків.

Діаграма, представлена вище, являє собою огляд всіх компонентів, які використовуються у різних версія TTFD, описаних раніше. Цікавим є опис архітектурного зразка, який використовується у різних версіях TTFD додатків. Фактично, вони використовуються один і той же шаблон: Model View Controller (MVC) (рисунок 19). Таким чином можна було розділяти основні компоненти TTFD, як вказано раніше написаних на C++, з користувацького інтерфейсу і конкретного коду додатку певної платформи.

Програмні додатки iOS та OS TTFD мають аналогічну структуру: є клас *MainViewController*, який є підкласом *NSViewController* для iOS, який керує статусом процесу рендеринга TTFD (і, в кінцевому рахунку, розрахунками CRI) і користувацьким вводом для налаштування процесу трасування променів.

Рисунок 19 – Архітектурний шаблон «Модель-Вигляд-Контролер»

Ці класи знаходяться в двох головних середовищах MVC, доступних на платформах Apple: Cocoa [42] для OS і XCocoa Touch [43] для iOS. Кожний з цих контролерів використовує Grand Central Dispatcher (GCD), інфраструктуру Apple для підтримки паралельного виконання коду і таким чином уникає блоку користувацького інтерфейсу [44]. Інтерфейс складається з:

- Користувацький інтерфейс для вводу;
- Користувацький інтерфейс для результатів рендеринга.

Ці компоненти користувацького інтерфейсу є особливою платформою, взятою з фреймворків користувацьких інтерфейсів, описаних раніше. Ця модель складається з:

- Всього C++ коду ядра TTFD. Він поширений між застосунками iOS і OS X з використанням вбудованого середовища Apple. Таким чином під час процесу збірки платформа може бути автоматично націлена на компіляцію на потрібній конкретній архітектурі платформи (x86-64-бітна Intel для OS X і armv7-amd64 для iOS).

- Власний клас растеризації для кожної платформи, який створює зображення за результатами даних рендеринга трасування променів, отриманих з основних класів TTFD, для відображення користувачу.

Версія TTFD для Windows 10 має аналогічну структуру. Існує клас *MainPage*, який є контролером додатку. Цей клас використовує задачу паралелізму для підтримки паралельного виконання коду і запобігає блокування користувацького інтерфейсу [45].

Користувацький інтерфейс створений з використанням XAML і власних UWP об'єктів операційної системи Windows 10. Модель складається з:

- Всього C++ коду ядра TTFD. Для Windows класи є включеними до складу головного проекту, оскільки є можливість включити їх в список компіляції самому прямо з головного проекту (x86 32-біт та 64-біт).
- Власний клас пастеризації як і для платформи Apple, використовує особливі функції Windows, щоб згенерувати зображення, що отримали у результаті рендеринга обраної сцени.

Версія командного рядка Linux не відповідає шаблону MVC. Фактично вона включає тільки модель: класи ядра TTFDE. Додаткові компоненти, необхідні для цієї версії – це основних метод, який є призначеним початком застосунку, і метод растеризатора, який використовується для генерації зображення рендерингової сцени. Отже, ці два методи входять до складу вихідного файлу main.cpp. Цей файл можна знайти у папці, яка включає всі основні класи ядра, але використовувати тільки у командному рядку TTFD. Синтаксис команди версії TTFD з командним рядком виглядає наступним чином: *./TestTracerforDiploma <scene filename path> <antialiasing> <width> <height> <result image name>*.

3.4 Інструменти розробки, тести і неперервна інтеграція

TTFD був розроблений з використанням інструментів, спеціалізованих для різних платформ. Це є наслідком того, що кожна операційна система, як було помічено раніше, використовує різні мови і має різні комп'ютерні архітектури.

Головні IDE для різних платформ, що були використані при розробці:

- XCode для Apple;
- Microsoft Visual Studio для Windows;
- CLion для Linux;

Всі версії TTFD мають набір тестових прикладів, написаних особливий фреймворків для різних платформ. Ці тестові приклади використовуються для неперервної інтеграції (continuous integration – CI): кожного разу, коли нова версія коду була перенесена у головну директорію проекту, всі тестові приклади для кожної платформи виконувалися і загальна збірка кожної версії програмного додатку створювалася для того, щоб процес компіляції працював належним чином. Платформи, що використовувалися для CI:

- Travis [46], для платформи Apple, тому що дає розробнику можливість використовувати одну налаштовану OS X віртуальну машину з останньою версією встановленого XCode. Саме тому є можливість генерувати різні варіації тестів для OS X та iOS (використовуючи симуляцію iOS);
- Appveyor [47], для Windows платформи, тому що це єдиний сервіс неперервної інтеграці, що підтримується Windows 10.

Процес встановлення і збірки версії TTFD з командним рядком був виконаний з використанням CMake, крос-платформного інструмента збірки. Фактично, версія командного рядка може бути легко перенесена на інші платформи, якщо файл CMakeList буде правильно налаштований.

РОЗДІЛ 4. PBR ТА CRI. ОЦІНКА ЕФЕКТИВНОСТІ РЕАЛІЗОВАНИХ МЕТОДІВ

У даному розділі будуть представлені деякі зроблені сцени. Вони покажуть всі особливості TTFD, описані раніше. Перші дві показані сцени будуть відображатись з використанням RGB даних і моделі Уайтеда, щоб показати типові зображенні, які можна отримати з допомогою стандартного механізму трасування променів. Інші сцени показали, як TTFD може відображати сцени на фізичному рівні, причому всі моделі BRDF були раніше.

4.1 Фізично обґрунтований рендеринг: реалізація деталей

Однією з головних особливостей TTFD є набір фізично обґрунтованих моделей BRDF, які він підтримує. Всі ці моделі, як була помічено раніше, являються спеціалізацією базового класу BRDF. Цей клас має властивість *brdfType,* яка вказує, який тип BRDF реалізує моделі. Кожна модель BRDF, визначена у TTFD, відноситься до одного з сімейств:

- Розсіюючий;
- Дзеркальний;
- Відбитий/віддзеркалений;
- Пропущений/заломлений.

Всі BRDF моделі реалізують три методи. Перший, *f(const Vecto& wi, const Vector& wo, const Intersection& intersection)*, реалізує рівняння моделі BRDF. Другий, *samplef(Vector* wi, const Vector& wo, const Intersection& intersection)*, реалізує рівняння моделі, як і попереднє, і відображає напрямок ω_i так, щоб деякі моделі трасування промені, такі як трасування шляху, могли використовувати його для трасування промені до джерела світла. Останній, *pdf(const Vector& wi, const Vector& wo, const Intersection& intersection)*, знову використовується у певних моделях, таких як трасування шляху, для вирішення рівняння рендеринга з використанням метода Монте-Карло. Останній буде згаданий під час пояснення

трасування і детально описано у пункті 4.1.3. У наступних параграфах повідомляється опис того, як різні моделі реалізовані у TTFD.

4.1.1 Розсіювальні та дзеркальні поверхні

Перша модель розсіювального сімейства також являється найпростішою: модель Ламберта. Як видно з рівняння, описаного у пункті 2.3.2, реалізація проста: спектр об'єкта ділиться на π. Для моделей Орена-Найяра і Торренса-Спарроу має бути використаний інший підхід. Їх рівняння в значній мірі базуються на використанні напрямку світла ω_i і ω_0 у сферичних координатах. Для їх обчислення зазвичай потрібне перетворення координат системи координат цього напрямку в дотичний простір з початком у точні перетину променів. Таким чином, обчислення сферичних координат може бути узгодженим для кожної точки перетину і для кожного напрямку.

Але іноді, коли це можливо, краще спиратися на тригонометрію і векторні операції, щоб уникнути затрат на калькуляцію перетворень. Ось чому ці моделі реалізовані з використанням операцій, які не використовують сферичні координати напряму. Як приклад, згадуючи формулювання Орена-Найара, член $\cos(\phi_i - \phi_0)$ може бути обчислений як добуток точок проекції двох кутів на дотичний простів з наступною формулою [48]:

$$\cos(\phi_i - \phi_0) = \|\omega_i - n(\omega_i \cdot n)\|\|\omega_0 - n(\omega_0 \cdot n)\| \quad (70)$$

Для виміряної моделі BRDF необхідний інший підхід. Як було описано у розділі 2.3.2.5, набори даних, який використовується в TTFD, є даними відбиття в Університеті Корнелла (Cornell University Reflectance Data). Ці набори даних індексуються з використанням сферичних координат. Ось чому, по-перше, потрібна генерація дотичної осі. Для осі і буде використовуватися нормаль у точці перетину поверхні. Для дотичних (*tangent*) і (*bitangent*) вектори доступні різноманітні підходи. Зазвичай система координат може створюватися строго з одного вектора,

скорочуючи одну із компонент і інвертуючи інші дві. Це було б гарною технікою для використання, оскільки виміряний BRDF, як і всі інші розсіювальні моделі, представлені в цій роботі, ізотропні, тому вони не потребуються орієнтації дотичної в конкретному напрямку. Цей корисний метод використовується у іншій частині TTFD, але для виміряного BRDF був вивчений інший спосіб: можна обчислити попередній добуток між нормаллю і напрямком ω_i для отримання варіанту дотичного вектору, а потім обчислити перетин (cross product????) між ним і нормаллю для отримання *bitangent*. Це бажаний метод, який TTFD використовує для обчислення дотичного простору для виміряної моделі BRDF і виконання зміни координатного перетворення у сферичні координати для напрямків ω_i і ω_0. Тепер потрібно зробити ще один крок: пряме зчитування прикладу з наборів даних, які знаходяться найближче до сферичних координат, отриманих з попереднього обчислення, не дуже корисного для рендеринга [38].

По цій причині необхідно використовувати конкретне перегрупування сферичної координати. TTFD використовує те, яке визначене Стівеном Машнером у його докторській дисертації [11]. Перегрупування задається наступною формулою (враховуючи, що у TTFD y це верхня вісь):

$$\text{remap}(\theta_i, \phi_i, \theta_0, \phi_0) = (\sin\theta_i \sin\theta_0 , \cos\theta_i \cos\theta_0 , \Delta\phi) \quad (71)$$

Це перегрупування має дві важливі властивості:

- Ізотропність BRDF завдяки $\Delta\phi$. Використовуючи цей член, пари напрямків (ω_i, ω_0), які мають різні значення ϕ, можуть бути пов’язані з одним і тим же набором вибірок, оскільки вони знаходяться на однаковій відстані $\Delta\phi$, навіть якщо ϕ не співпадають. Таким чином BRDF не залежить від дотичного напрямку, який використовується для перетворення ω_i і ω_0;
- Враховує взаємність BRDF.

Після перегрупування дані виміряної BRDF можуть бути використані під час рендеринга.

4.1.2 Модель дзеркального відбиття/пропускання

Останні реалізовані моделі у TTFD це моделі відбиття та пропускання.

Вони використовуються разом для створення такого скляного матеріалу. Важливі реалізовані деталі мають бути описані. Клас *MateriaBRDF*, який використовується для створення матеріалів з використання фізично обґрунтованих моделей BRDF, має два основних методи: *f(const Vector3D& wi, const Vector3D& wi, const Intersection& intersection)* і *samplef(Vector3D* wi, const Vector3D& wo, const Intersection& intersection, const BRDFType type)*. В першому враховуються всі BRDF, пов’язані з матеріалом і додається їх вклад. Другий метод представляє конкретний тип BRDF і дає SPD матеріалу і новий напрямок ω_i для трасування. Для всіх інших типів BRDF перший метод чи другий може бути використаний для отримання загального SPD матеріалу в залежності від використовуваного типу трасувальника. Але це не відноситься до моделі дзеркального відбиття/пропускання: всі типи моделей трасування променів, які підтримує TTFD, використовують другий метод, оскільки новий напрямок має бути завжди заміряний (sampled), щоб враховувати і відбиття і заломлення, викликані цими BRDF моделями.

4.2 Трасування шляху

У пункті 2.2 даний загальний опис алгоритму трасування шляху. Зокрема, інтеграція Монте-Карло була вказана як спосіб знайти числове рішення інтегралу, який знаходиться у рівнянні рендеринга.

Методи інтеграції Монте-Карло – це потужні математичні інструменти, які можна використовувати для наближення значення інтегралу. Як це працює? Перед тим, як давати свій опис, необхідні деякі визначення.

1. Випадкова змінна X - це змінна, обрана деяким випадковим процесом. Вона може бути дискретною або неперервною.
2. Кумулятивна функція розподілу (Cumulative Distribution Function, CDF) $P(x)$ випадкової величини – ймовірність того, що вона прийме значення, яке менше чи рівне деякому значенню з її розподілу:

$$P(x) = Pr\{X \leq x\} \tag{72}$$

3. У випадку неперервності випадкової величини функція щільності ймовірності (probability density function, PDF) дає ймовірність того, що випадкова величина прийме конкретне значення. Зазвичай воно визначається як добуток CDF (див. рівняння 73) і зазвичай інтегрується до 1 по своїй області:

$$p(x) = \frac{dP(x)}{dx} \tag{73}$$

4. Очікуване значення $E_p[f(x)]$ функції f є його середнім значення у порівнянні з деяким розподілом значень. Воно визначається як (з D, який відповідає області функції):

$$E_p[f(x)] = \int_D f(x)\, p(x) dx \tag{74}$$

5. Дисперсія функції – відхилення від її очікуваного значення.

Зараз можливо дати визначення методу інтеграції Монте-Карло: заданий набір однорідних випадкових величин $X_i \in [a, b]$, очікуване значення оцінювання Монте-Карло F_N, представлене у рівнянні 75, дорівнює інтегралу функції над інтервалом $[a, b]$ [49].

$$E[F_N] = E\left[\frac{b-a}{N}\sum_{i=1}^{N} f(X_i)\right] = \int_a^b f(x)dx \tag{75}$$

Коли випадкові змінні нерівномірні, PDF необхідно приймати до уваги і оцінювати:

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \quad (76)$$

Як це відноситься до трасування шляху і як це працює в деталях?

Трасування шляху обчислює N вибірок для кожного пікселя. Кожна вибірка представляє собою колір, отриманий при трасуванні всіх відбиттів променя від камери до джерела світла (або до максимальної кількості відбиттів, і у цьому випадку колір чорний, або SPD рівне 0 у випадку спектральних даних).

Напрямок відбиття променя, яке виникає, коли промінь потрапляє в об'єкт, представляє собою тільки новий промінь, обраний у відповідності з конкретним розподілом. Потім всі кольори, отримані з різноманітних вибірок, усереднюються. Результатом цього процесу є очікуване значення для оцінки по методу Монте-Карло, яке відповідає одному з рівнянь рендеринга, представленому у розділі 2.1.

TTFD використовує загальну реалізацію трасування шляху з додатком: розподіл, з якого обраний новий промінь, не є однорідним і обирається з розподілу з формою, аналогічною формі розподілу BRDF. Цей метод називається важливою вибіркою. Фактично, TTFD використовує:

- Косинусний зважений розподіл (cosine weighted distribution) для всіх розсіювальних BRDF;
- Розподіл з подібністю з розподілом Блінна для BRDF Торренса-Спарроу.

Ключовою особливістю трасування шляху TTFD є інформація, яка використовується для оцінки кольору: всі розрахунки виконуються з використання спектрального розподілу потужності для предметних матеріалів і джерел світла. Потім значення, отримане для кожного пікселя, перетворюється у значення тристимула, а потім відображається колір sRGB. Щоб регулювати загальну потужність світла у сцені,

використовувався параметр яскравості. Також використовується гамма-корекція sRGB, як і для іншої моделі, яка входить до TTFD, яка використовує спектральні дані, щоб краще декодувати кольори, отримані зі спектральних даних.

Таким чином, трасування шляху TTFD є допустимим кандидатом, який буде використовуватися при оцінці точності кольору на візуалізованій сцені і корисний для візуалізації феномену світла, такого як метамеризм (навіть якщо останній можна було побачити з моделлю Уайтеда). Очевидно, що рендеринг сцени з трасувальником шляху є найбільш вражаючим, який може генерувати TTFD.

4.3 Приклад 1. Сцена RGB з використанням моделі Уайтеда

Хоча головний фокус TTFD сконцентрований на спектральному рендерингу, він також дозволяє відображати стандартну RGB сцену з використанням емпіричної моделі освітлення.

Було зроблено і відображено дві сцени з 4-кратним згладжуванням (antialiasing). Перша сцена рендерилася з використанням моделі Фонга (рисунок 20), а друга – з використанням Блінна-Фонга (рисунок 21). Є об’єкти, які складаються з різних матеріалів: розсіювальних, відбиваючих, скляних і дзеркальних.

Рисунок 20 – RGB сцена: модель Уайтеда, Фонга, рельєф

Рисунок 21 – RGB сцена: Уайтеда, Блінн-Фонг, різноманітні текстури

TTFD підтримує, тільки для RGB рендеринга, деякі методи відображення текстур.

Процес накладання текстур був вперше розроблений Едвіном Катмулом в його докторській дисертації у 1974 році [50]. Текстура представляє собою 1D, 2D або 3D масив пік селів. Текселі – це зразки текстур. Вони класифікуються наступним чином:

- Кольоровий тексель, який містить дані RGBA. Текстура, яка складається з кольорового текселя, представляє собою кольорову карту;
- Альфа тексель, який містить альфа-значення;

- Тексель нормалі, який містить значення напрямків нормалей. Текстура, яка складається з текселя нормалі, можу бути картою нормалей або картою рельєфу.

Дві зображені RGB сцени містять різни типи текстурного мапування.

Перша сцена показує реалізацію методу мапування куба. Він був винайдених Недом Гріном у 1986 році [51].

Даний метод використовує шість кольорових мап, які застосовуються до кубу, який включає всю візуалізовану сцену. Ця кольорова мапа використовується для представлення вигляду світу у всіх напрямках. Промені, трасовані у сцені, які не перетинають який-небудь тривимірний об’єкт, відображають колір з однієї з кольорових мап з використанням базової алгебри. Фактично, компонент з найбільшою величиною вказує, яка з шести сторін куба повинна бути використана. Інші два компоненти використовуються для вибірки кольорової мапи. З допомогою цього методу був створених Skybox, який містить першу сцену.

Друга сцена показує приклад процедурної текстури, згенерованої з використанням шуму Перліна. Процедурна текстура – це текстура, яка генерує «на льоту» тексель для конкретної позиції. Шум Перліна – це алгоритм, який можна використовувати для створення процедурної текстури. Він був винайдений Кевіном Перліном у 1983 році [52][53].

Головна ідея заключається у об’єднанні псевдовипадкових шумних сигналів для генерації природних випадкових ефектів (наприклад, хмар чи лави). Синя/біла сфера у верхній частині куба і оранжева/червона сфера в центрі сцени рендеряться з використання цієї техніки.

Друга сцена також показує приклад зображення рельєфу. Bamp mapping – метод, винайдений Джеймсом Блінном у 1978 році [54] для моделювання грубих поверхонь. Під час обчислення освітлення нормалі відновлюються з карти рельєфу і використовуються для зміни сприйнятого світла. Нормаль також може бути згенерована процедурно з використанням шуму Перліна. Рубінова сфера в правому нижньому куті є

прикладом цієї методики (з використання шуму Перліна для генерації карти нормалей процедурно).

4.4 Приклад 2: спектральні сцени з використанням моделі Уайтеда

У цьому прикладі представлені три сцени. Вони рендеряться з використанням моделі Уайтеда, спектральних даних та 4-кратним згладжуванням.

Всі обчислення кольору зроблені з використанням 81 прикладу для кожного використовуваного SPD: відстань між кожною довжиною хвилі дорівнює 5 нм.

Перша сцена містить всі фізично обґрунтовані BRDF, які підтримуються у TTFD: Ламберта, Орена-Найяра, дзеркального відбиття і заломлення, Торренса-Спарроу і виміряна BRDF. Сцена рендериться з використанням різних джерел світла, щоб побачити ефект кожного з них з різним набором кольорів.

На рисунку 22 тільки що описана сцена рендериться з використанням джерела світла D65. Кольори у цій сцені виглядають природно тому, що, як була помічено раніше, це джерело світла імітує свічення на відкритому повітрі, яке можна порівняти з сонячним світлом. На рисунку 23 одна і та ж сцена рендериться з використанням освітлення FL4. Як можна легко помітити, колір блакитної сфери, створений з використанням BRDF Торренса-Спарроу, і сфери крилонового синього, створені з виміряною BRDF, відрізняються від попередньої сцени завдяки SPD FL4.

На рисунку 22 показана та ж сама сцена, але цього разу рендерилася з використанням освітлення FL9. Це джерело світла є частиною підмножини сімейства F, які виглядають як флуоресцентна лампа повного спектру світла. Ця підмножина включає в себе освітлення FL7 та FL8. Ці джерела світла моделюють люмінесцентні лампи, які наближають ввесь спектр видимого світла.

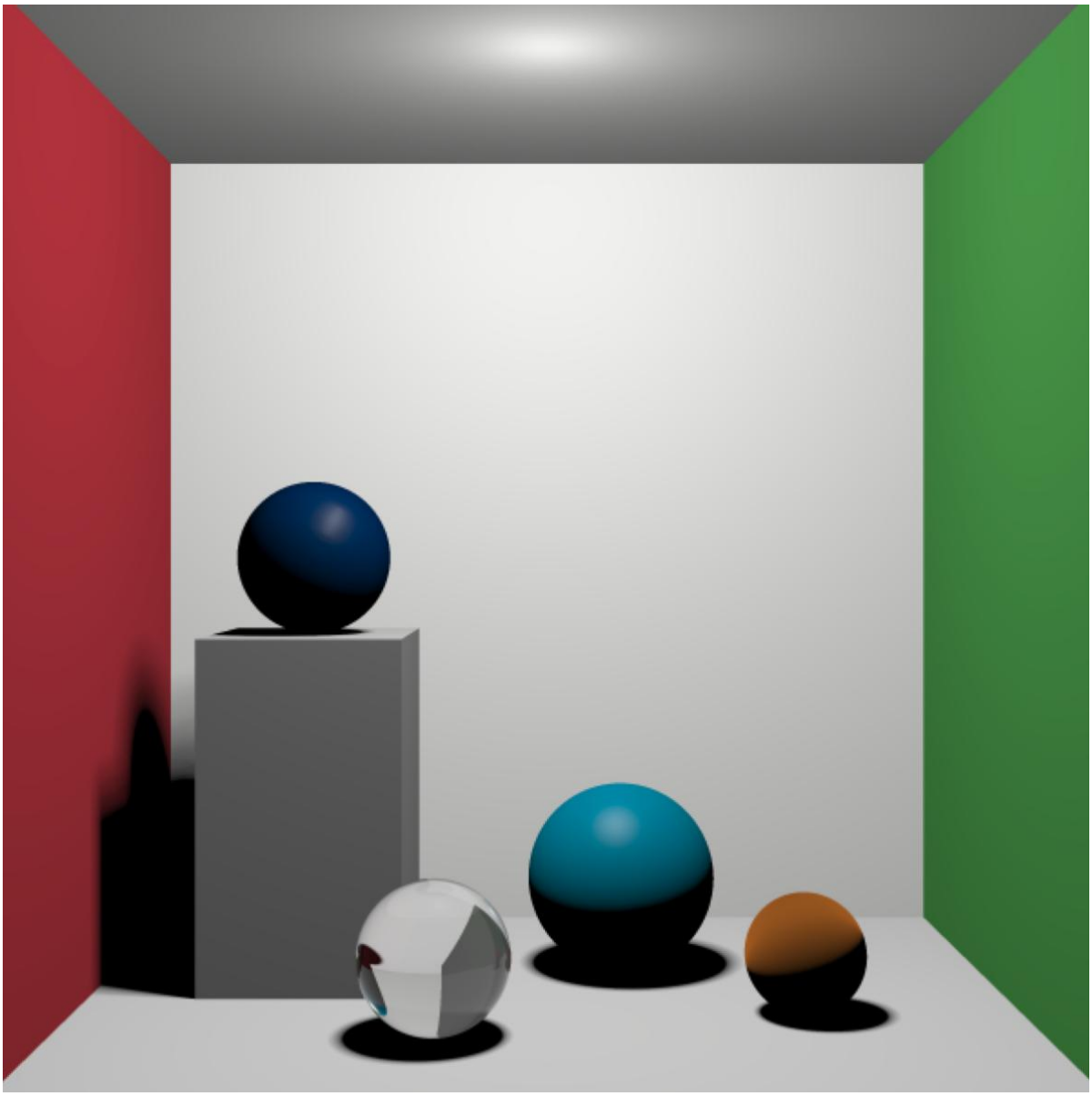

Рисунок 22 – Спектральна сцена: Уайтед, фізична обґрунтована BRDF, джерело світла D65

По цій причині лампи з SPD такого типу мають високий CRI і вони вважаються одним одними з найточніших джерел світла по відношенню до природного видимого світла. Насправді, як видно, результат рендеринга набагато більш схожий на результат, показаний на рисунку 22. На рисунку 25 знову показана одна й та ж сцена востаннє, яка рендериться зі стандартним джерелом світла A. Як вже було відмічено, це джерело світла імітує типове внутрішнє джерело світла з розжареною вольфрамовою ниткою.

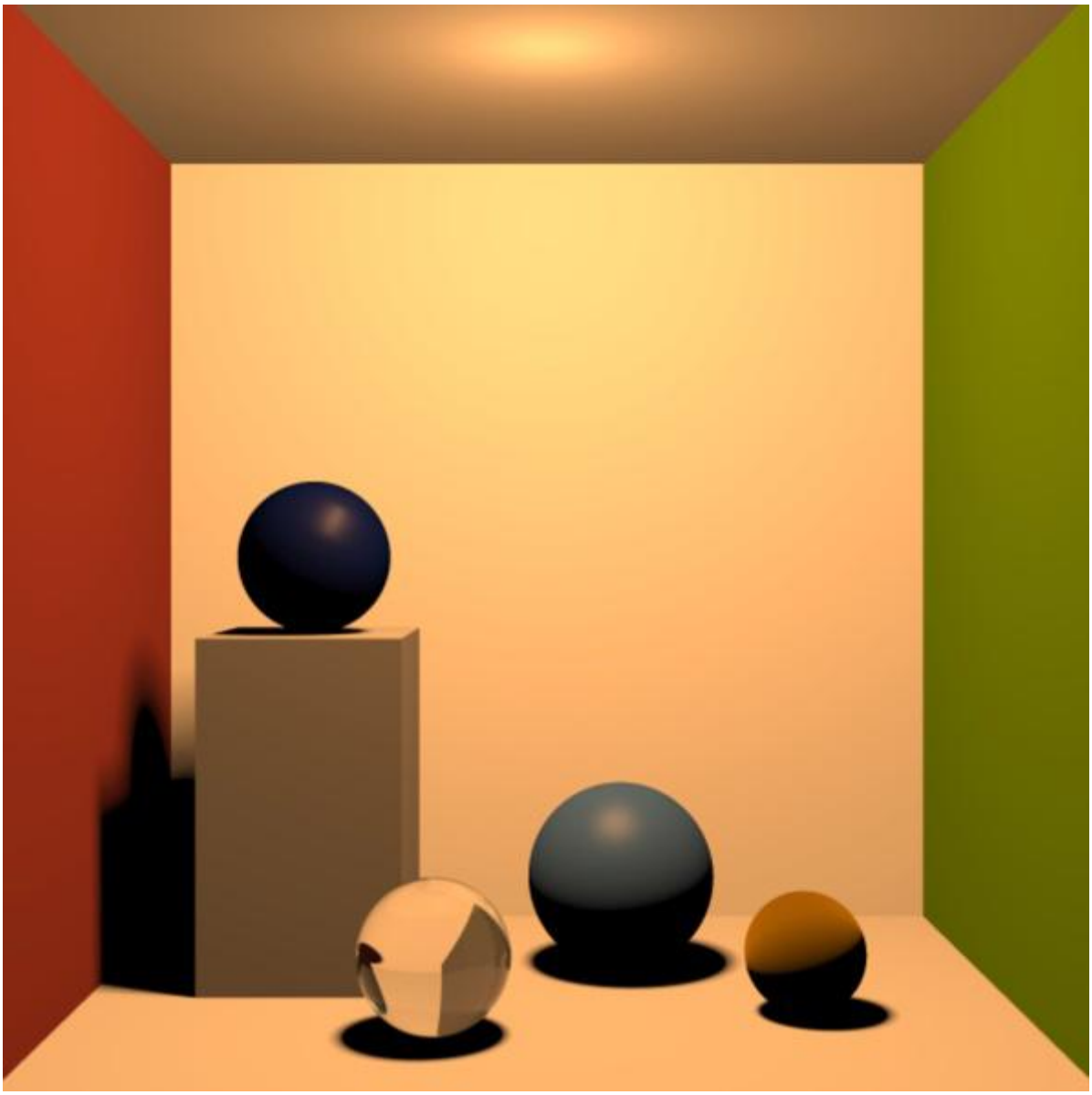

Рисунок 23 – Спектральна сцена: Уайтед, фізична обґрунтована BRDF, джерело світла FL4

Ось чому це зображення є цікавим: те, що воно зображує, ця сцена так, наче освітлена звичайною лампою. Легко побачити, як колір впливає на SPD освітлення, а кольорове зображення деяких з візуалізованих об’єктів відрізняється від їх оригіналу з освітленням D65.

Друга сцена, представлено в цьому дослідженні, показує всі типи виміряних BRDF, доступних у TTFD. Як вказано у розділі 2.3.2, набір даних, який використовується у TTFD, є даними коефіцієнта відбиття Кернелла. Існує чотири виміряних BRDF: крилоновий синій, найманський, гранатовий червоний та акриловий синій. Рисунок 26 містить сцену з цими

виміряними BRDF, які рендеряться за допомогою джерел світла, які використовувалися у попередній сцені.

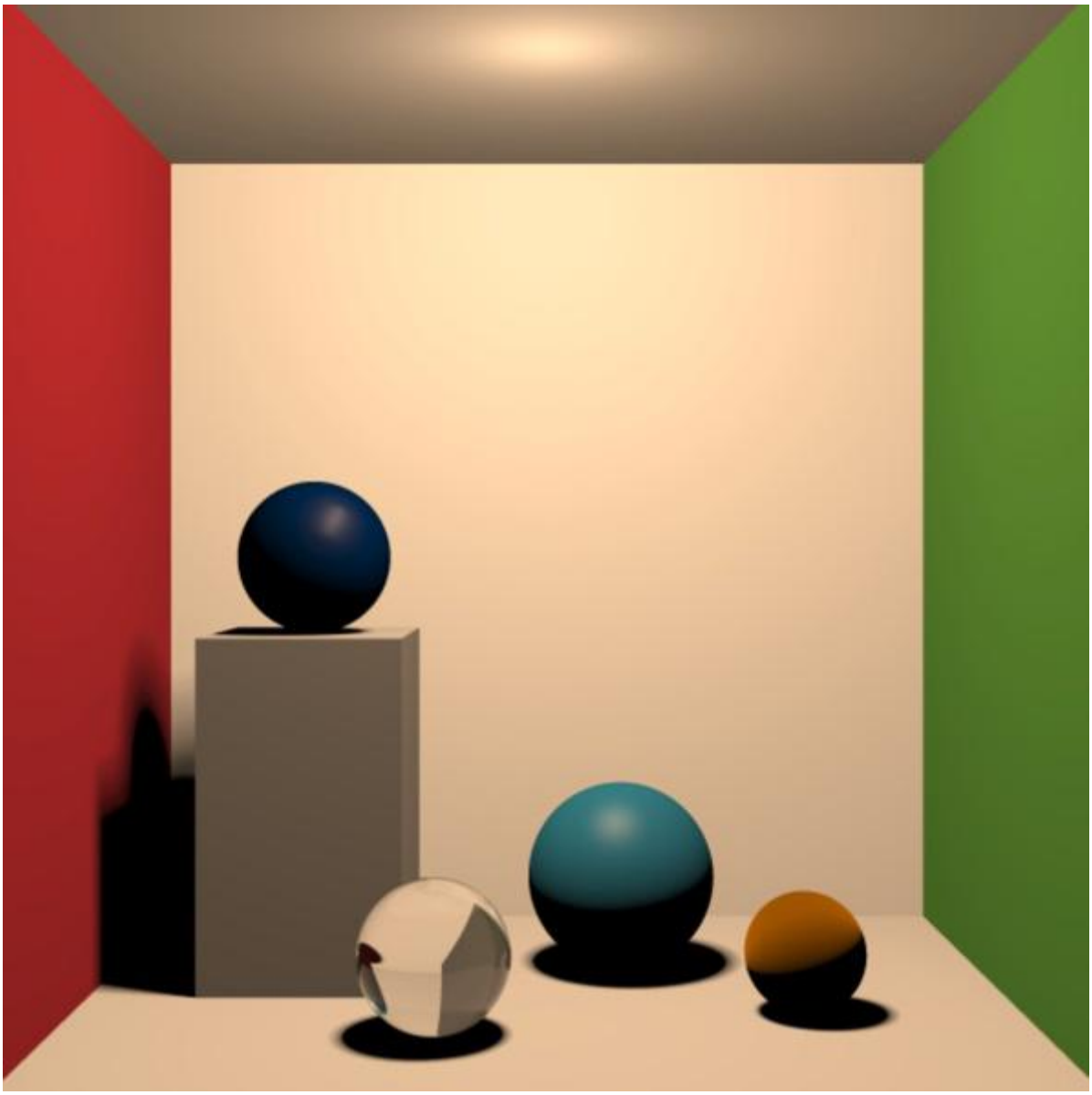

Рисунок 24 – Спектральна сцена: Уайтед, фізична обґрунтована BRDF, джерело світла FL9

У третій сцені показана ще одна особливість TTFD, яка не строго корелює з PBR: можливість рендеринга полігональної сітки. Багатокутна сітка представляє з себе набір пов’язаних багатокутників (породжених з вершин та ребер), яка визначає форму моделі. Найбільш розповсюдженим полігоном вважається трикутник.

На рисунку 27 показана сцена з усім BRDF Лабмерта, яка містить модифіковану версію 3D-модель Стендфордського дракону, який

знаходиться у Стендфордському репозиторії 3D-сканування [54]. Вихідна сітка зі Стендфордського репозиторію складається з 871414 трикутників. Це значить, що для кожного променя всі ці трикутники мають бути перевірені, щоб пересвідчитись, чи є перетини, що приводять до низької продуктивності у швидкості рендеринга.

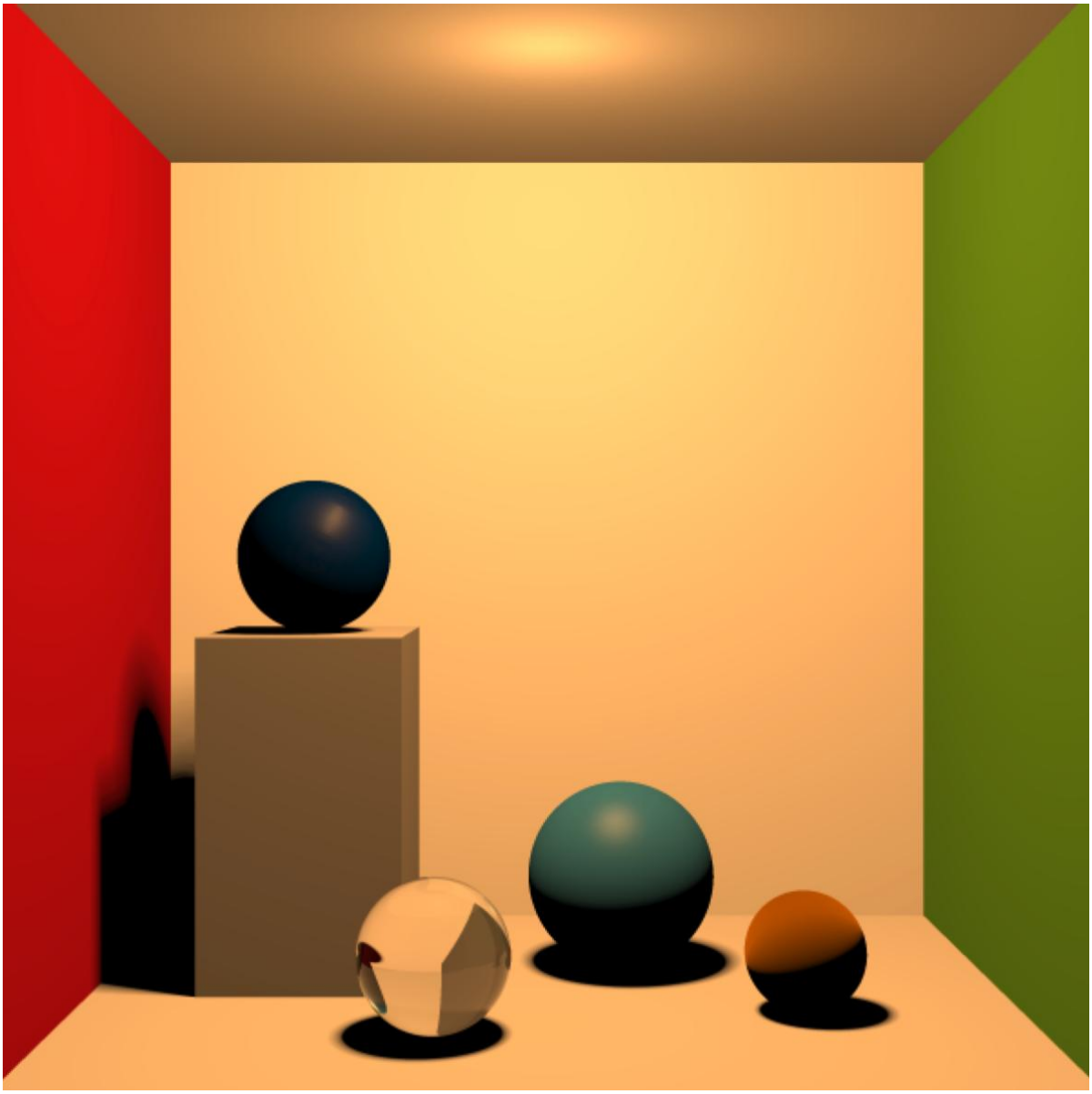

Рисунок 25 – Спектральна сцена: Уайтед, фізична обґрунтована BRDF, джерело світла A

Це пов’язано з тим, що у стандартному алгоритму трасування променів тест перетину променів-об’єктів має складність $O(n)$, де n - кількість об’єктів, які підлягають трасуванню.

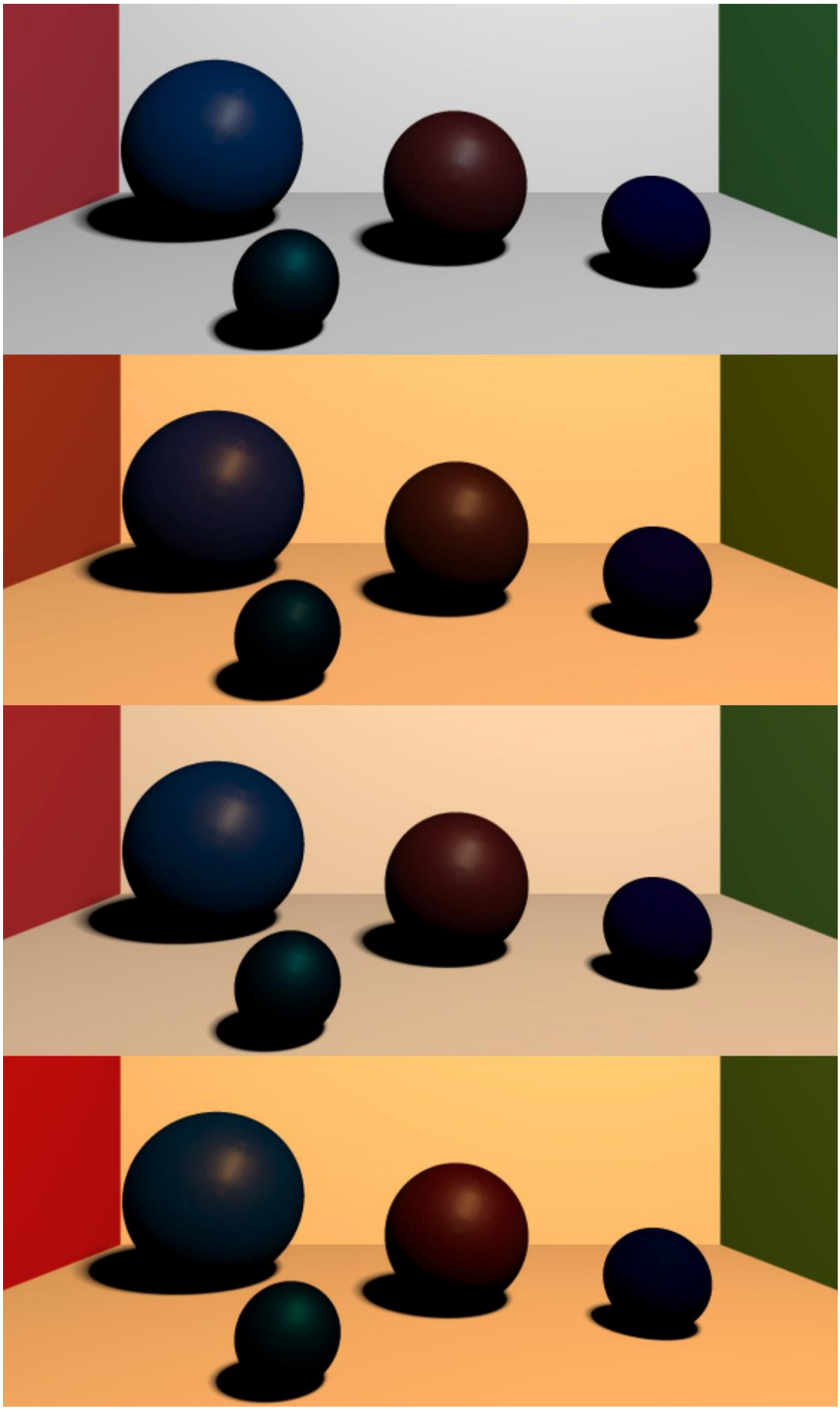

Рисунок 26 – Виміряні дані BRDF, джерела світла D65, FL4, FL9, A

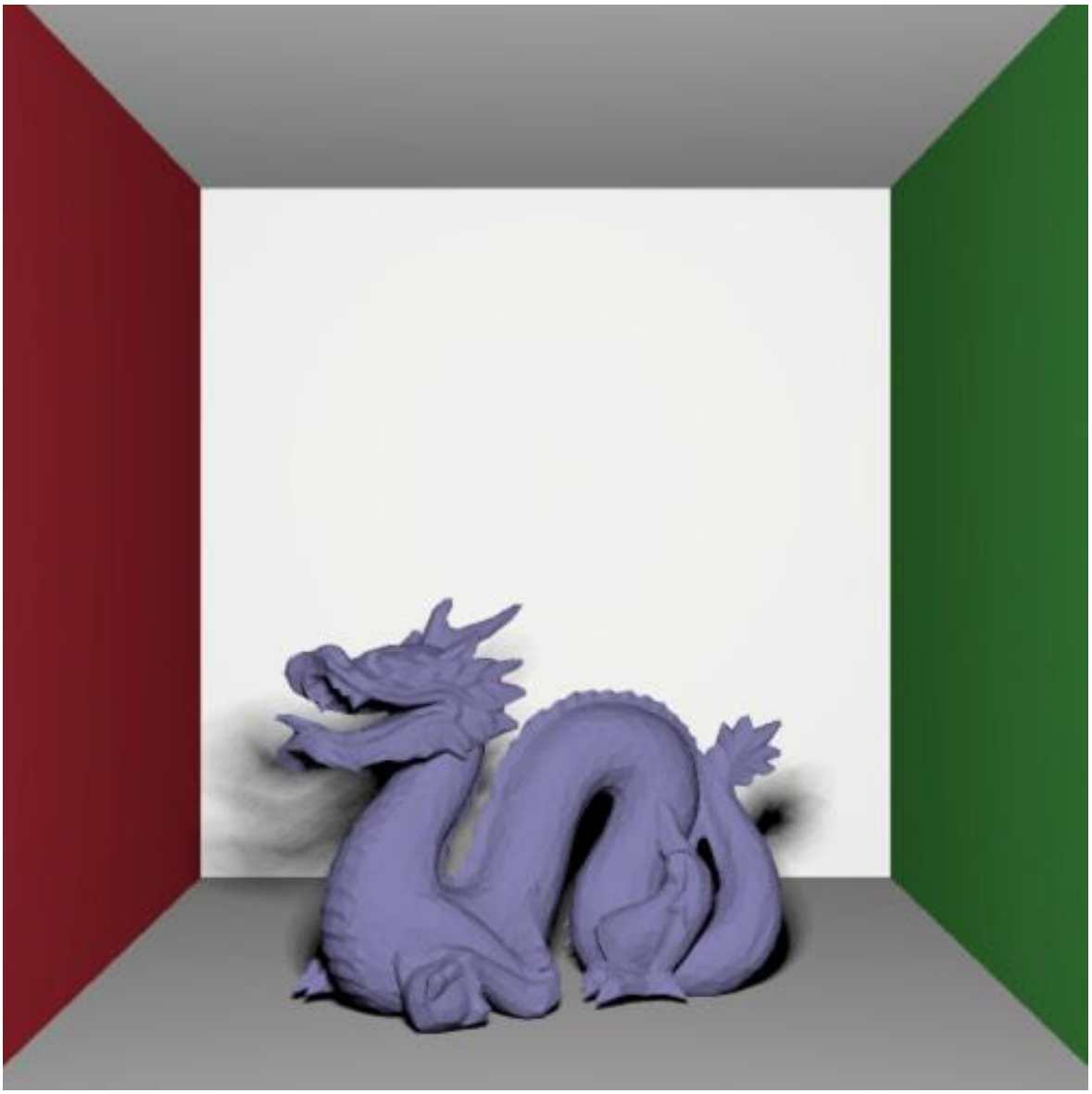

Рисунок 27 – Спектральна сцена: Уайтед, сітка (ламбертіан), джерело світла D65

Щоб покращити продуктивність рендеринга сітки, оригінальний Стендфордський дракон був спрощений з використанням Meshlab [56]. Таким чином, сітка, яка використовується у сцені, складається лише з 7423 полігонів, але зберігає гарний рівень деталізації в порівнянні з вихідною моделлю. Спрощена сітка була недостатньою для забезпечення розумної продуктивності. Ось чому TTFD підтримує техніку прискорення процесу перетинання променів: обмежений по осі обмежувальний прямокутник (AABB) [57].

Полігональна сітка обмежена в боксом (ящиком), вирівняному у світових координатах сцени. Під час тесту перетину променів замість перевірки всіх полігонів сітки виконується перший тест: промінь перевіряється на перетин з обмежуючим боксом. Якщо він не перетинається, то промінь не перетинає жодного з багатокутників сітки, тому їх можна пропустити під час тесту перетинання. Перетин боксів обчислюється за загальним рівнянням перетину променів. Унаслідок того, що бокс суміщений з віссю системи координат, рівняння перетину площини і променя спрощується, щоб покращити швидкість його обчислення.

4.5 Приклад 3: спектральні сцени з використанням трасування шляху

Це тематичне дослідження пов’язане з однією з ключових особливостей TTFD: спектральне трасування шляху. Всі зображення рендеряться з використання 2-кратного згладжування і, як для спектральних сцен Уайтеда, використовується 81 приклад для кожного SPD.

Представлені різні зображення, які містять рендеринг різних сцен з використанням цієї моделі. Варто відмітити рівень реалізму об’єктів у сценах:

- Всі скляні сфери відображають каустику на найближчих поверхнях;
- Всі об’єкти показують природні м’які тіні по замовчуванню, без необхідності в артефактах або додаткових специфічних променях.

Як правило, щоб отримати сцену повністю очищену від шуму, необхідні тисячі прикладів, просто після всього лиш 50 прикладів зображення стає упізнаваним, а після 4000 сцена набуває гарного рівня різкості.

Перша представлена сцена містить дві скляні сфери, одну оранжеву розсіюючи сферу, отриману з використанням BRDF Орена-Найяра, і одну блакитну розсіюючи сферу, отриму з використанням BRDF Ламберта.

Вона представлена у декількох версіях, основаних на кількості прикладів: 20 (рисунок 28), 200 (рисунок 29), 800 (рисунок 30), 4000 (рисунок 31) і 20000 (рисунок 32). В цій сцені немає об’єктів, які використовують BRDF Торренса-Спарроу.

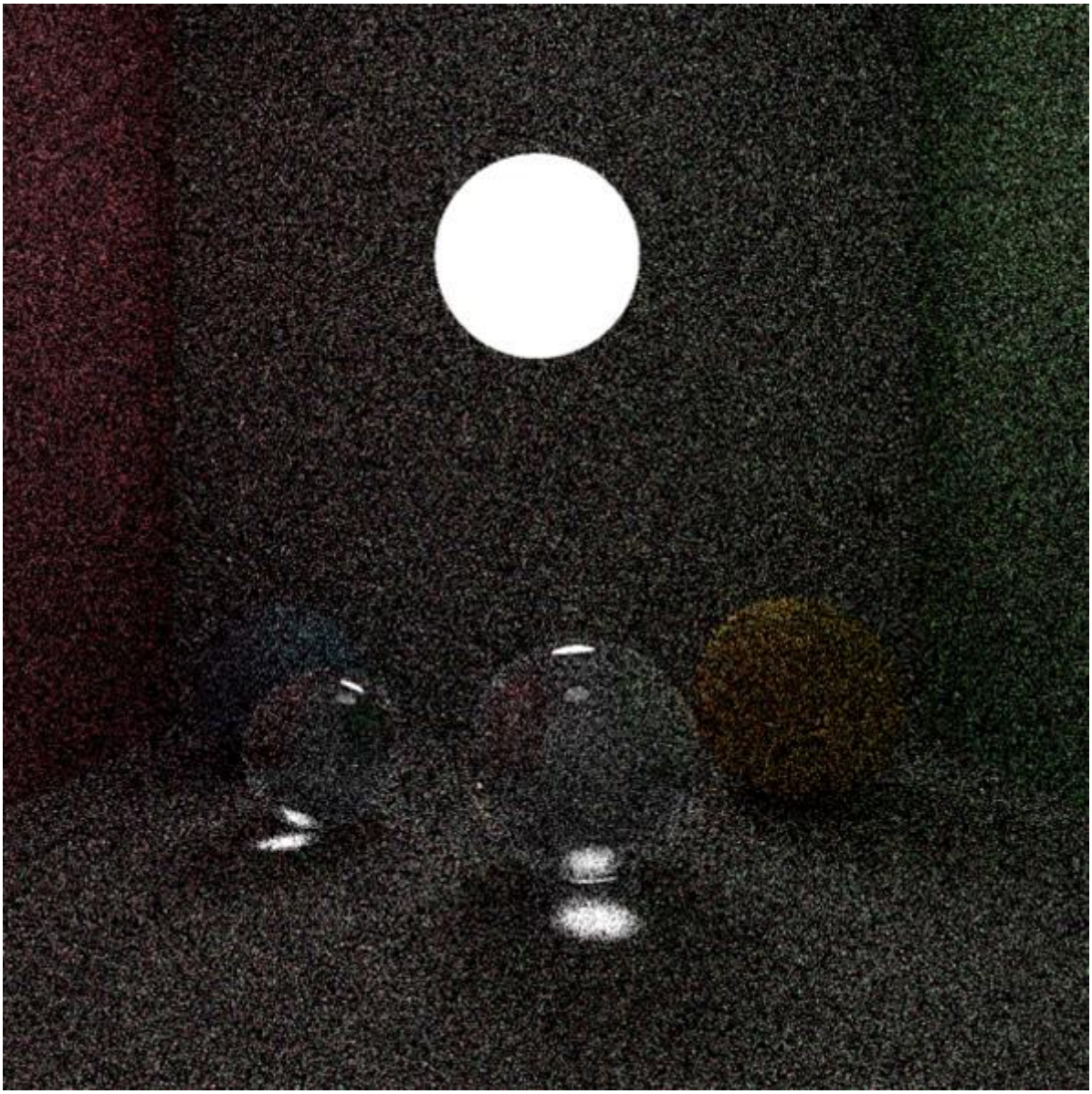

Рисунок 28 – Перша спектральна сцена з трасуванням шляху: 20 вибірок на піксель, джерело світла D65

Це пов’язано з тим, що за допомогою трасування шляху BRDF Торренса-Спарроу отримує неймовірний рівень реалізму. Внаслідок того, що це BRDF з відбиваючими мікрогранями, глянцеві ефекти з’являються, коли сфера рендериться за допомогою трасування шляху. Щоб пресвідчитися, що отримана сфера правильна, була створена нова сцена:

одна сфера у центрі боксу Корнелла з чорно-білим спектром Макбета відповідно для розсіюючи і дзеркальних компонентів. Сцена з аналогічними налаштуваннями була створена також з використанням найвідомішого фізично обґрунтованого механізму рендеринга, приведеного у вступі: PBRT [7].

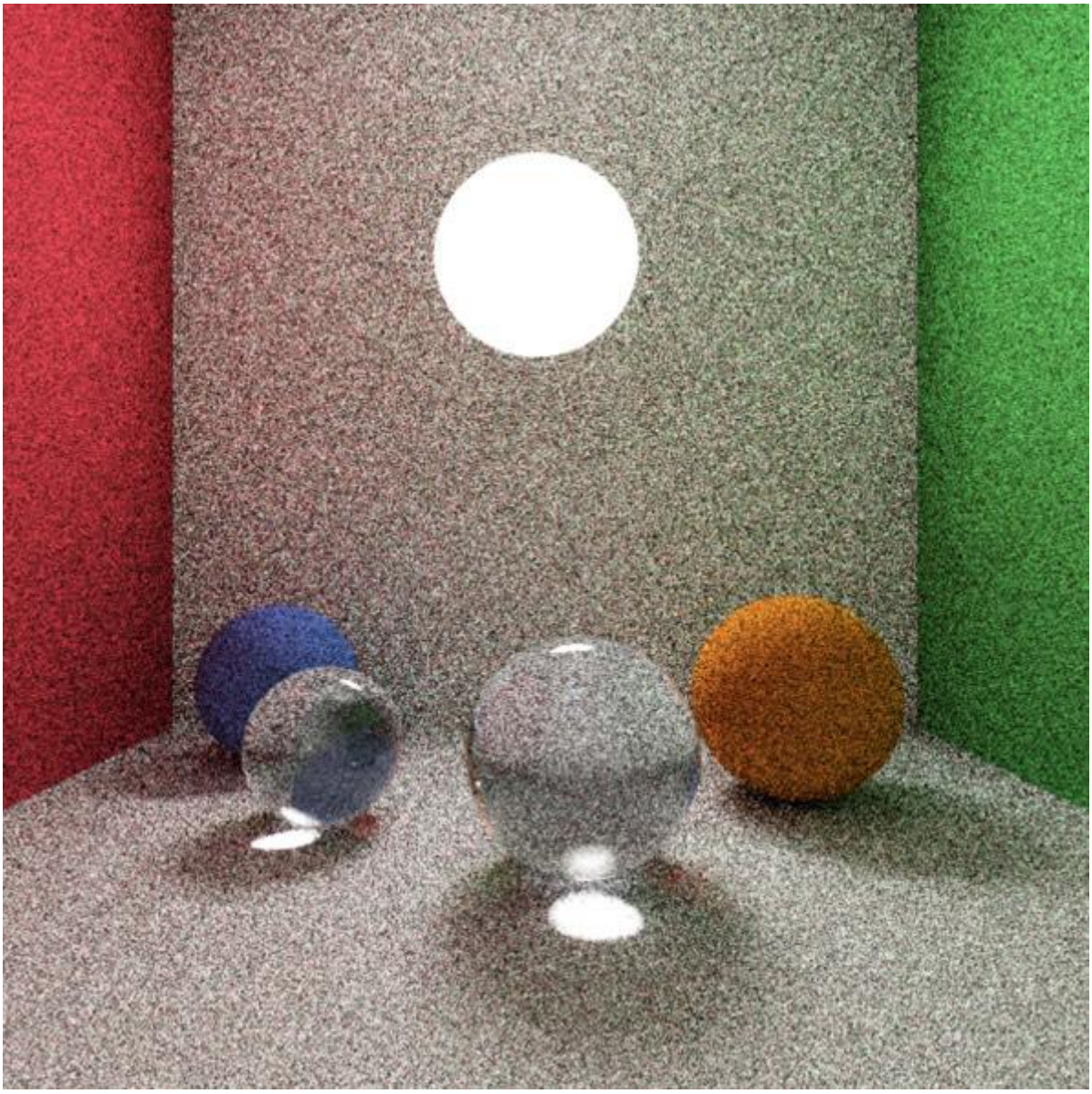

Рисунок 29 – Перша спектральна сцена з трасуванням шляху: 200 вибірок на піксель, джерело світла D65

Єдина різниця у двох сценах: спектр, який використовується для частини боксу Корнелла, нейтральний 8 на сцені TTFD і білий у PBRT, колір світла, визначений з використанням спектру D65 у TTFD і з простими RGB значеннями у PBRT.

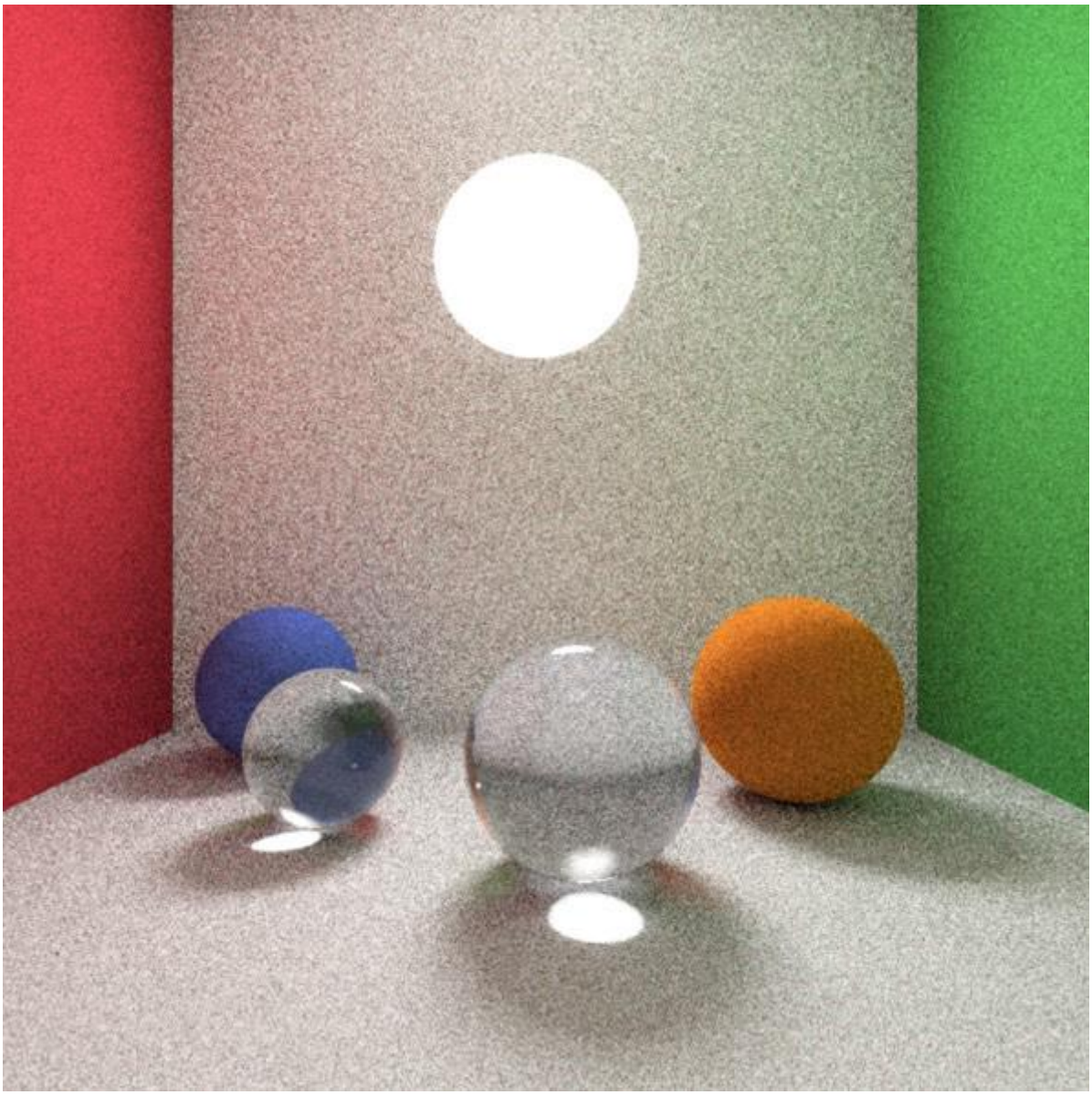

Рисунок 30 – Перша спектральна сцена з трасуванням шляху: 800 вибірок на піксель, джерело світла D65

Варто відмітити, що PBRT використовує «російську рулетку» і інші методи для рендеринга сцени якнайшвидше і з гарним рівнем різкості з використанням декількох зразків. Таким чином у цьому прикладі PBRT використовує 1000 зразків, але TTFD потребує 10000 зразків, щоб отримати сцену з аналогічним результатом.

Як бачимо із зображення, отриманого на рисунку 33, TTFD зобразив сферу Торренса-Спарроу так, як і PBRT. Глянцевий ефект на сфері чітко видно, при цьому стінка боксу Корнелла відображається на правильній стороні.

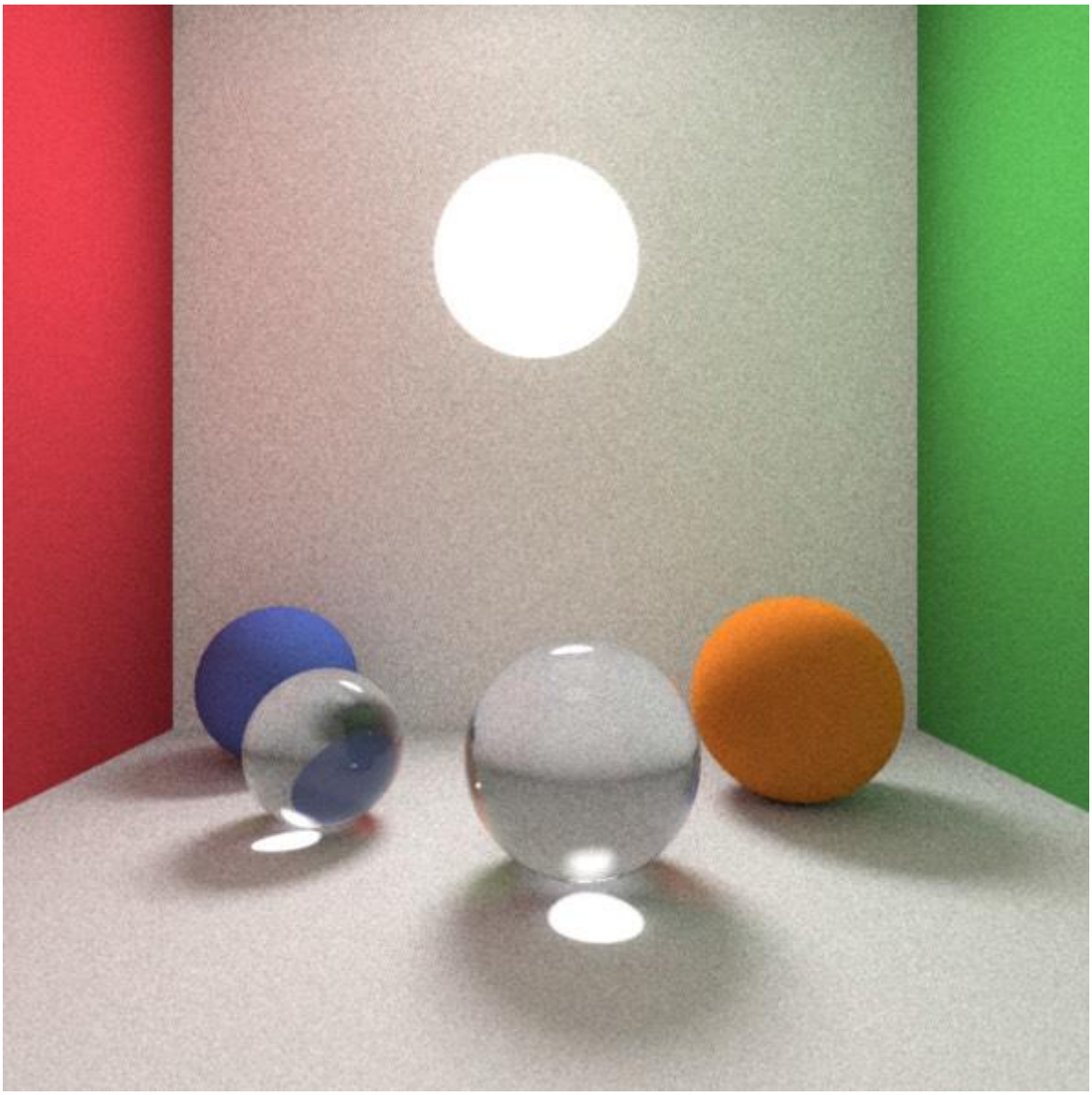

Рисунок 31 – Перша спектральна сцена з трасуванням шляху: 4000 вибірок на піксель, джерело світла D65

На рисунку 34 показаний результат. Сцена рендерилася з використанням 20000 зразків і D65 у якості світла. Легко побачити рівень реалізму сцени. Всі сфери виглядаються більш природно. Їх тіні втрачають фальшивий вигляд, який воно отримали у моделі Уайтеда. Сфера Торренса-Спарроу набуваю реалістичного глянцевого ефекту (той же вигляд сцени, що і на рисунку 30), синя сфера з виміряною BRDF виглядає більш однорідною, а дзеркальна підсвітка більш визначена і, нарешті, скляна сфера більш пов’язана відносно навколишнього простору.

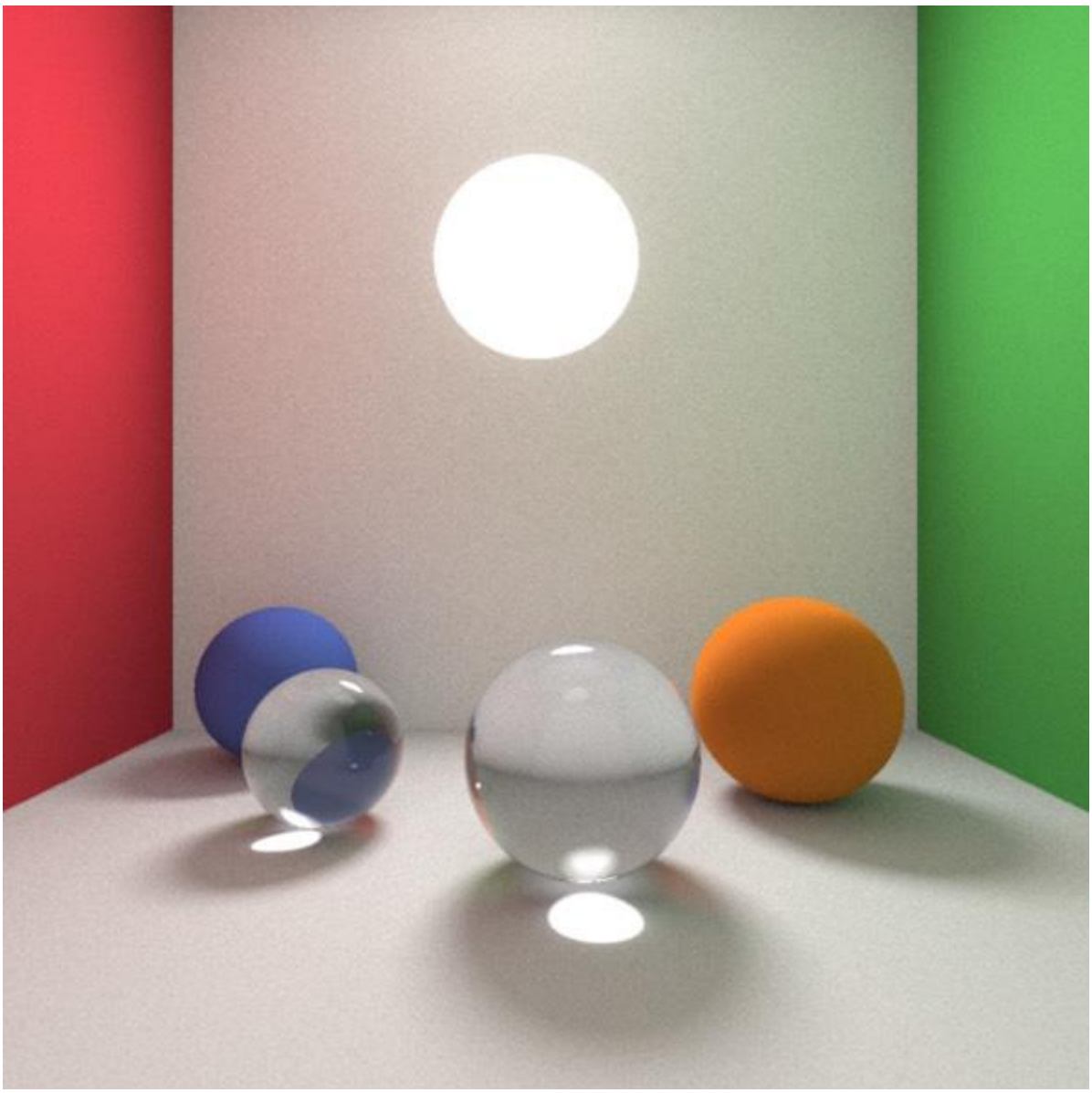

Рисунок 32 - Перша спектральна сцена з трасуванням шляху: 20000 вибірок на піксель, джерело світла D65

На рисунку 35 та ж сцена рендерилася з використанням спектральним трасуванням шляху і освітлення FL4. Результати у кольорі співпадають з використанням моделі Уайтеда.

По всьому шляху трасування джерело світла відображається як об’єкт всередині нього. Це дозволяє користувачу TTFD генерувати сцену з більшими візуальними ефектами, змішуючи вихідне світло об’єктів всередині сцени з іншими. На рисунку 36 показаний приклад, в якому

сцена відображається з джерелом світла, розміщеним всередині нього та низьким рівнем яскравості.

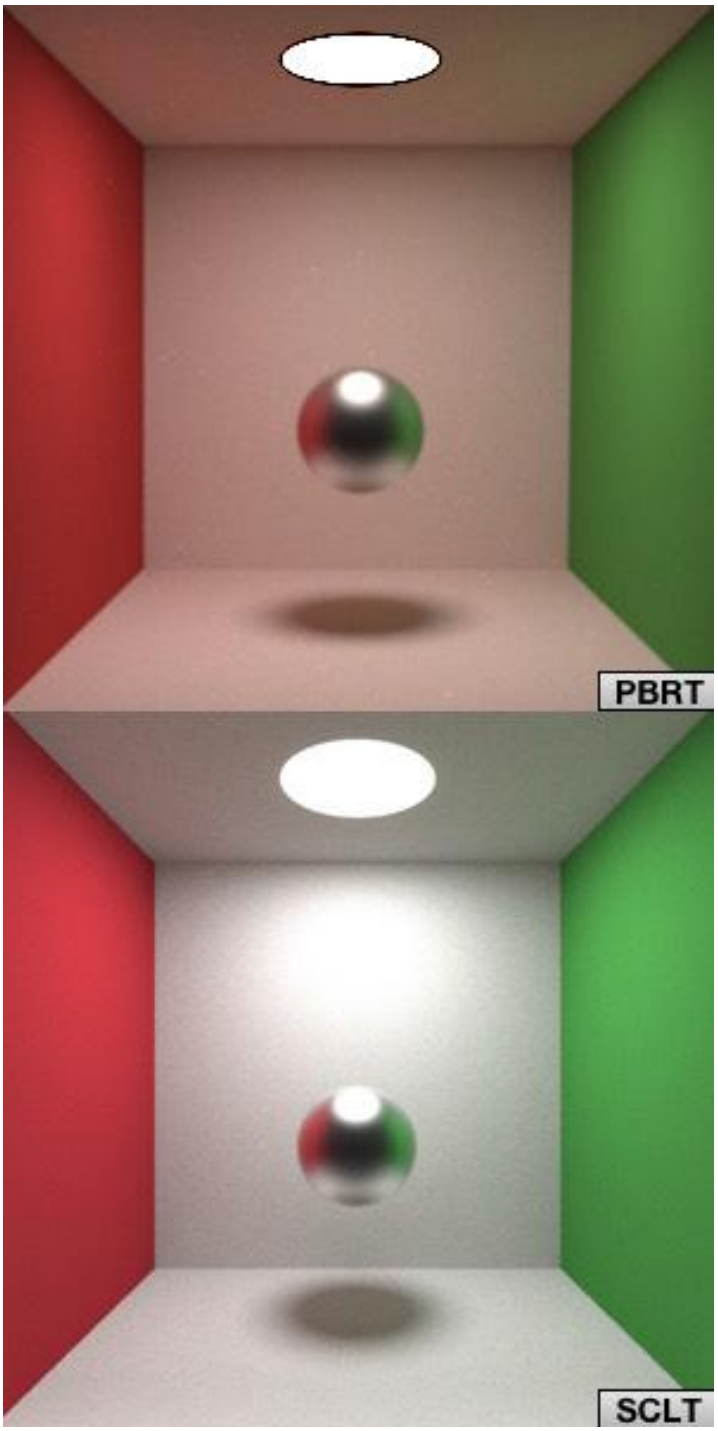

Рисунок 33 – Сцена Торранса-Спарроу: PBRT 1000 вибірок, TTFD 1000 вибірок

Інша сцена представлена на рисунку 37. Вона містить новий бокс Корнела з сумішшю BRDF (у тому числі з гранатовою червоною виміряною BRDF) з освітленням FL9.

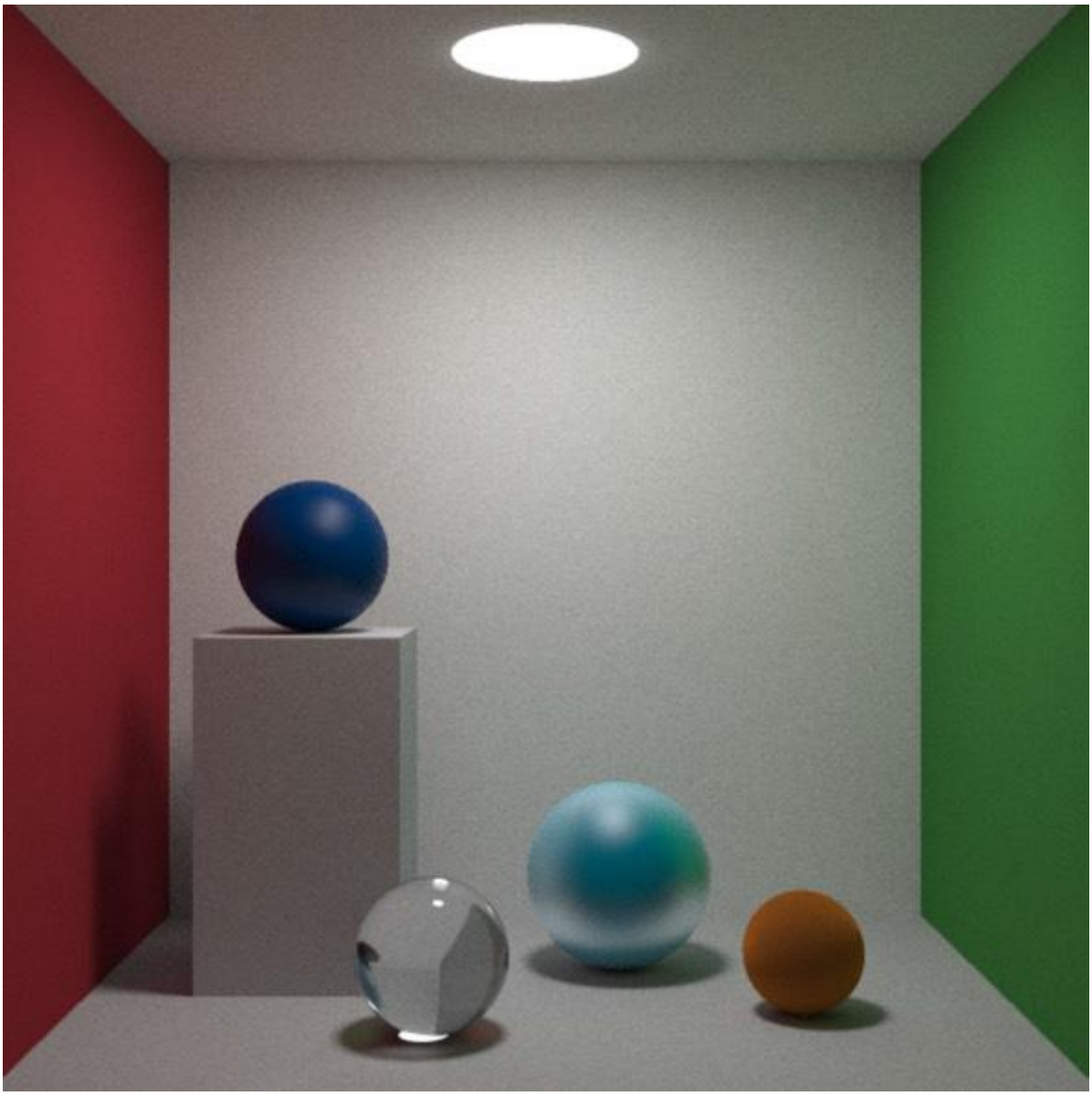

Рисунок 34 – Сцена навчальної вибірки 2 з трасуванням шляху: 20000 вибірок, джерело світла D65

Останній представлений у попередньому параграфі сценарій трасування шляху містить як останню сцена, показану у попередньому абзаці, так і полігональну сітку дракону з репозиторія сканування Стендфорда [55]. Це було зроблено із застосуванням різних BRDF: Ламберта на рисунку 38, Торренса-Спарроу на рисунку 39 і дзеркальне відбиття та пропускання на рисунку 40.

Рисунок 35 – Сцена навчальної вибірки 2 з трасуванням шлаху: 20000 вибірок, джерело світла FL4

Рисунок 36 – Спектральна сцена 2 трасування шляху з джерелом світла D65 всередині: 20000 вибірок

Рисунок 37 – Спектральна сцена 2 трасування шляху: 20000 вибірок на піксель, джерело світла FL9

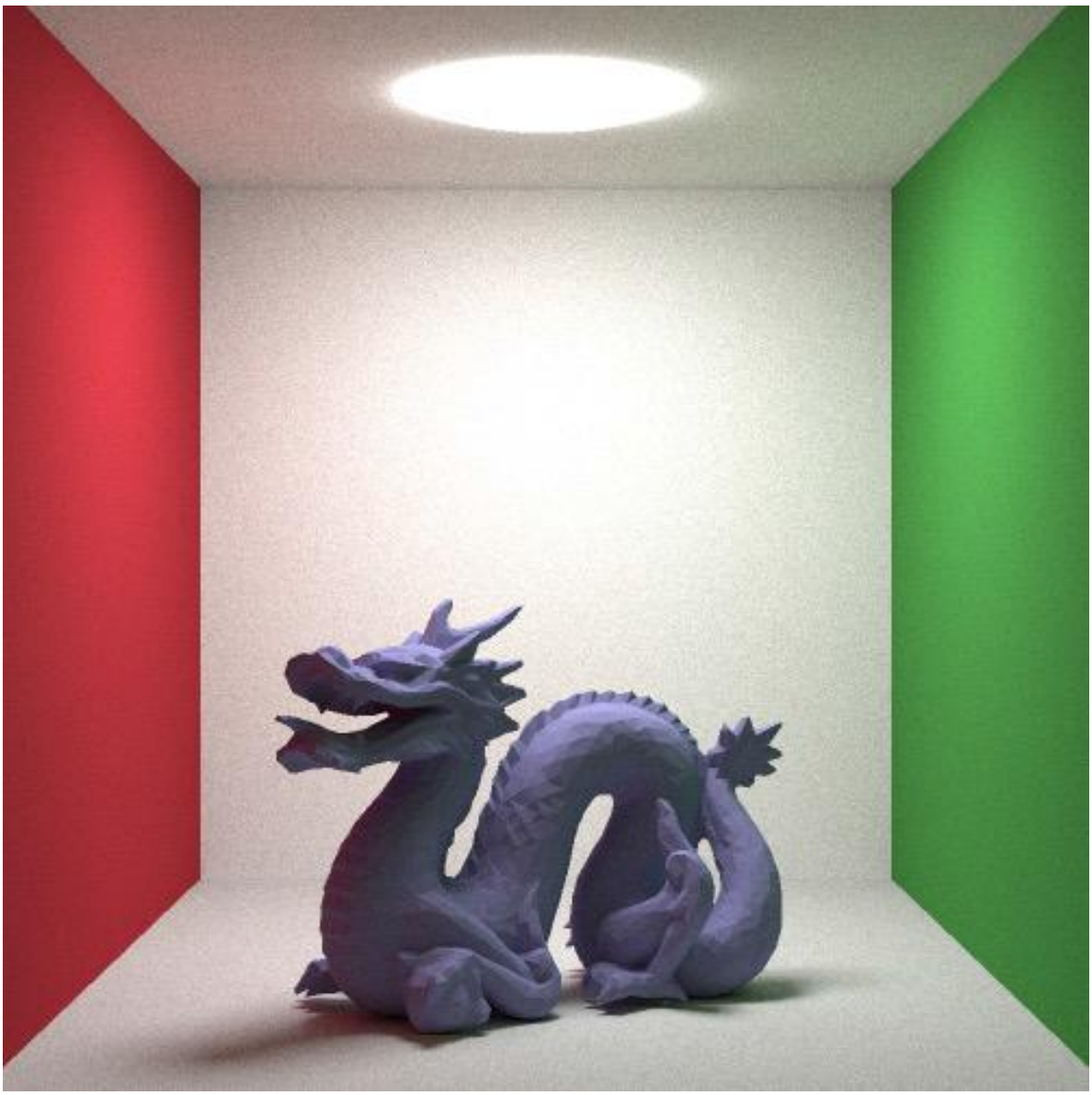

Рисунок 38 - Спектральна сцена трасування шляху: сітка (ламбертіан), 10000 вибірка

Рисунок 39 – Спектральна сцена трасування шляху: сітка (Торранс-Спарроу), 10000 вибірок

Рисунок 40 – Спектральна сцена трасування шляху: сітка (відбиття і заломлення)

4.6 Індекс передачі кольору

Як повідомлялося у першому розділі, другий закон Грассмана описує явище, яке називають метамеризмом, тобто можливість того, що один і той же колір можна було б сприймати з об’єктів з різними SPD, коли вони освітлюються одним і тим же джерелом світла, який має певний SPD. Цей факт головним чином корелює з використанням штучних джерел світла.

Як можна оцінити здатність джерела світла виявляти всі частоти його кольорового спектру у порівнянні з контрольним світлом? Відповідь проста: індекс передачі кольору (CRI).

Навіть якщо він потрапив під критику і CIE намагається замінити його іншими індексами, наприклад Color Quality Scale (CQS) і CIECAM02, він використовується у стандартних комерційних програмах для оцінки точності кольору джерела світла і зазвичай вказується на комерційних світлових продуктах. У цій роботі CRI буде використовуватися по-іншому. Він буде використовуватися для оцінки точності, з якою джерело світла буде відтворювати колір у сцені, створеній з використанням TTFD.

CRI визначається у діапазоні від 0-100: джерело світла з CRI 100 буде відтворювати колір з точністю, подібній тій, що має природне світло.

TTFD підтримує обчислення CRI, використовуючи два різних методи: метод пробних зразків і R96a. CIE визначив CRI як [58]:

«Вплив джерела світла на колір об’єктів шляхом свідомого чи підсвідомого порівняння з їх кольоровим зовнішнім виглядом під контрольним освітленням»

4.7 Метод пробних зразків

Метод тестових зразків – стандартна процедура для розрахунку CRI.

У ньому порівнюється передача кольору тестового джерела з кольором, отриманим від ідеального світла, яке базується на CCT джерела світла, яке підлягає тестуванню. Метод використовує зразки, взять з атласу Манселла. Кроки, необхідні для отримання CRI, позначені через R_a, є [15]:

1. Перетворити SPD джерела світла в значення тристимула з допомогою 2° стандартного спостерігача;
2. Перетворити значення тристимула, отримані у попередньому кроці, в хроматичні координати CIE 1931 XYZ;
3. Обчислити координати кольорового простору CIE 1960 (хроматичні координати) використовуючи координати CIE 1931 XYZ, отримані у попередньому кроці;
4. Обчислити CCT тестового джерела. Це можна обчислити з Планківського локуса у координатах кольорового простору CIE

1960. Для цілей цієї роботи CCT буде обчислений за допомогою апроксимуючого алгоритму МакКамі за формулою:

$$CCT = -449n^3 + 3525n^2 - 6823.3n + 5520.33 \quad (77)$$

де n це:

$$n = \frac{x - 0.3320}{y - 0.1858} \quad (78)$$

5. Оберіть згадане джерело світла: якщо CCT менше 5000 К, то використовуємо закон випромінення Планка (рівняння 10) для оцінки SPD чорного тіла, у іншому випадку використовувати стандартне освітлення D65;
6. Перетворити SPD зразків у значення тристимула, використовуючи тестове джерело світла і еталонне джерело світла;
7. Застосувати CAT Вон Кріеса до кожного зразку, отриманому на попередньому кроці;
8. Перетворити зразки, адаптовані до Вон Кріеса, в кольоровий простір CIE 1964;
9. Обчислити Евклідову відстань ΔE_i: між парою координат одного і того ж зразку під тестом та посиланням;
10. Обчислити для кожного зразку спеціальний CRI, використовуючи наступну формулу:

$$R_i = 100 - 4.6\Delta E_i \quad (79)$$

11. Знайти загальний CRI R_a, обчисливши середнє арифметичне спеціальних CRI.

4.8 Метод $R96_a$

$R96_a$ - це оновлений CRI, описаний у попередньому абзаці. CIE почав працювати над цим у 1991 році і опублікував його у технічному комітеті 1-33. Цей показник так і не дістався до реального застосування у промисловості через розбіжності між дослідниками і індустрією (промисловістю). Для цієї дисертації цікаво розрахувати індекс $R96_a$ для

простого факту: він використовує SPD з 10 кольорів, взятих з перевірки Macbeth у якості зразку для розрахунку. Ці SPD однакові для рендеринга об'єктів, які містяться у різних сцена, представлених раніше. Кроки, необхідні для розрахунку $R96_a$ [16]:

1. Перетворити SPD тестового джерела світла у значення тристимула за допомогою 2° стандартного спостерігача і перетворити його координати у хроматичні координати CIE 1931 XYZ;
2. Обчислити CCT тестового джерела;
3. Обрати контрольне джерело світла з наступного списку, виходячи з мінімальної різниці з CCT тестового джерела світла: 2700K, 2950K, 3450K або 4200K випромінювання чорного тіла, D50 (5003K), D65(6504K);
4. Перетворити кожний зразок SPD, який тестується і знаходиться під контрольним джерелом світла в значення тристимула з використанням 2° стандартного спостерігача;
5. Застосувати CIECAT94 для адаптації кожного зразку, який тестується і знаходиться під контрольним джерелом світла, до D65;
6. Перетворити попередньо отримані зразки у CIE L* a* b* кольоровий простір;
7. Обчислити для кожного зразку в CIE L* a* b* спеціальний CRI за наступною формулою:

$$R_i = 100 - 3.248\Delta E_i \tag{80}$$

8. Знайти загальний CRI $R96_a$, обчисливши середнє арифметичне спеціальних CRI.

4.9 CRI: деталі реалізації

Як TTFD обчислює CRI?

Реалізації включає клас *ColorRenderingIndex*. Спектр джерела світла передається у дві функції:

1. *testSampleMethod(Spectrum<constant::spectrumSamples> spectrumTestLight)*
2. *r96aMethod(Spectrum<constant::spectrumSamples> spectrumTestLight)*

Ці два методи відповідають двом раніше описаним процесам розрахунку CRI: методу тестовим зразків і методу $R96_a$.

Оскільки ці методи потребують різних кольорових просторів, описаних у параграфі 2.1, клас для кожного з них був реалізований з різними методами перетворення. Ці класи:

1. *CIELab*
2. *CIEUCS*
3. *CIEUVW*
4. *CIW1931XYZ*

Приклад обчислення CRI

Як було помічено раніше, TTFD по замовчуванню підтримує різні джерела світла, в тому числі, джерела світла A, сімейство F, D50, D65 для того, щоб користувач міг рендерити сцени і виконувати обчислення CRI для джерела світла з різними SPD. Ось чому деякі тести були проведені для оцінки різниці між двома раніше описаними метода розрахунку CRI і точністю TTFD відносно очікуваного значення CRI для джерел світла, які підтримуються по замовчуванню.

Що стосується розрахунку кольору у фізичному спектральному рендерингу, то всі обчислення були виконані з використанням 81 зразку для кожного SPD, взятого з Munsell чи з перевірки кольору Macbeth.

Очевидним є те, що так, як TTFD підтримує RGB сцени/рендеринг, то перевірка виконується у випадку, якщо користувач обчислює CRI у спектральній сцені.

Перший набір перевірених джерел світла – сімейство F. Це не випадковий вибір: у цьому сімействі є джерело світла FL4, яке використовується Міжнародною Комісією по захисту навколишнього

середовища для калібрування індексу передачі кольору у методі тестових зразків. У наступній таблиці показані значення, отримані TTFD для двох методі і очікуване значення зразків тестового метода.

Таблиця 2 – Індекс передачі кольору джерел світла сімейства F

Джерело світла	**Метод тестових зразків CRI – очікуване значення**	**Метод тестових зразків TTFD CRI**	**TTFD CRI R96a**
FL1	76	74.67	69.75
FL2	64	64.64	69.73
FL3	57	57.23	64
FL4	51	50.92	57.60
FL5	72	64.12	58.32
FL6	59	58.49	55.53
FL7	90	90.31	88.91
FL8	95	88.13	96.13
FL9	90	90.48	89.74
FL10	81	79.77	81.27
FL11	83	83.04	75.19
FL12	83	83.22	78.26

Як видно з даних, TTFD в основному має високий рівень точності, якщо проводити порівняння між очікуваними значенням методу тестових зразків CIE і розрахованим значенням TTFD.

Можна відмітити тільки невідповідність освітлення FL5 та FL8. Це може бути враховано в даних SPD, які використовуються для цих двох джерел світла, можливо не настільки точних. З цієї ж таблиці варто відмітити, що метод R96a генерує переривчасті значення з невідповідностями практично всьому методу тестових зразків CRI.

Той факт, що цей метод має більш складне обчислювальне навантаження, ніж метод тестових зразків, завдяки використанню CIECAT94, а також той факт, що CIE ніколи не схвалював його в якості стандарту, залишає його реалізацію просто «іграшковим експериментом» для оцінки і порівняння зі стандартним методом тестових зразків. Варто також відмітити, що для CIECAT94 потрібне велика кількість параметрів, значення яких складно отримати. TTFD встановлює ці значення по замовчуванню. Наприклад, адаптована яскравості тестового поля визначається для того, щоб мати ту ж величину адаптованого освітлення опірного поля.

Таблиця 3 – Індекс передачі кольору для джерел світла A та D65

Джерело світла	**Метод тестових зразків CRI – очікуване значення**	**Метод тестових зразків TTFD CRI**	**TTFD CRI R96a**
A	98	99.74	86.52
D65	100	100	100

У таблиці 3 показані CRI для джерел світла A та D65. Знову реалізація TTFD методу тестових зразків являється дуже точною. Нарешті D65 дає значення 100, оскільки від представляє SPD природного світла, враховує максимальних рівень точності передачі кольору, а значення CRI помічає цей факт.

Рисунок 43 показує і приклад обчислення TTFD CRI. На рисунку 44 вся спектральна сцена, представлена за допомогою моделі Уайтеда, показаний їх CRI, отриманий TTFD.

ВИСНОВКИ

Рендеринг, який ґрунтується на законах фізики – це сучасне і майбутнє (можна тільки майбутнє залишити).

TTFD показує, як розглянуті методи можуть покращити зображення, отримані у рушії трасування променів. Головна мета рушія – рендерити сцени PBR з використанням спектральних даних для отримання максимального рівня точності передачі кольору.

Фактично, використання значень тристимула в якості основи для обчислення кольору дає TTFD можливість досліджувати явища, подібні метамеризму, які важко показати з стандартною обробкою RGB кольору. Можливість обчислення CRI для джерела світла дає користувачу можливість оцінити загальну точність відтворення сцени.

Можливість рендеринга PBR сцен з використанням спектральних даних і підтримкою обчислення CRI дозволяє TTFD стати ідеальним інструментом для промислового дизайну і виробництва освітлення. Фактично, TTFD можна використовувати для якісної (навряд чи кількісної) властивості світлового продукту показувати кольори. Підтримка декількох пристрої і операційних система дає користувачу свободу вибору бажаної платформи.

Навіть якщо в ході цієї дисертації були продемонстровані деякі вражаючі особливості, TTFD далеко не ідеальний 3D-рушій. Можна було б додати і вивчити велику кількість покращень, щоб дозволити TTFD стати справжнім рушієм, корисним не тільки дослідницькій сфері, але і у виробничій. Зокрема, головною цільовою областю, яка має бути покращена, є:

- Швидкість та продуктивність рендеринга;
- Додаткові BRDF моделі;
- Джерела світла.

Що стосується швидкості та продуктивності, на даний момент TTFD працює повільно.

Фактично, для отримання зображення за допомогою трасування шляху з рядом зразків, подібних до вказаних в попередніх розділах, TTFD потребує багато часу для обчислень. Покращення в даній області можуть бути двох типів.

Перший з них пов'язаний з реалізованими моделями трасування променів, зокрема трасування шляху. На даний момент трасування шляху TTFD є стандартною версією цього алгоритму, яка не враховує який-небудь явний прямий підрахунок світла. Інші рушії, такі, як згаданий PBRT або smallPT в одному зі своїх варіантів [59], враховують цей вид обчислення. Для зміни стандартного алгоритму трасування шляху можна додати ще одну техніку, російську рулетку.

Другий тип пов'язаний з тестом перетину шляхів. Це вказано у пункті 4.3. Ця частина рушія трасування променів є найскладнішою для обчислення машиною. Фактично, для кожного променя тест перетинання має складність $O(n)$, де n - кількість об'єктів, які підлягають тестуванню. Щоб покращити цю частину, для прискорення процесу можна використовувати деяку структура даних, яка називається структурами даних прискорення:

- *k-d* дерево, структура даних для організації сцени в ієрархію просторових множин;
- ієрархія обмежуючих томів, структура даних для організації сцени в ієрархію примітивних множин.

Ці структури можуть привести тест перетинання до складності $O(\log n)$.

Друга область покращення пов'язана з моделями BRDF, які підтримуються в TTFD: на даний момент існують тільки ізотропні BRDF. Можна реалізувати додаткові анізотропні моделі, такі, як, наприклад,

BRDF Уарда, введені Григорієм Дж. Уардом у 1992 році [60] і BRDF Ашихміна, представленою Майклом Ашихміном і Пітером Ширлі у 2000 році [61]. Таким чином, TTFD зможе створювати більш широкий набір поверхонь (наприклад, матові метали).

Третя область покращення пов’язана зі світлом. Перш за все, в даний момент TTFD не реалізує ніякої вибірки світла, як, наприклад, PBRTE. TTFD також використовує емпіричний параметр яскравості для налаштування загальної яскравості світла у сцені для моделі трасування шляху. Це може бути замінено більш реалістичним розрахунком, заснованому на різноманітних характеристиках джерел світла, такого як площина та потужність.

Навіть якщо деякі з представлених покращень можуть поліпшити загальний досвід користування TTFD, він залишається гарною відправною точкою для тих, хто цікавиться вивчення рендеринга, який базується на законах фізики та колориметрії.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Tiffen Mfg. Color Temperature & Color Rendering Index DeMystified [Електронний ресурс] / Tiffen Mfg.. – 2014. – Режим доступу до ресурсу: http://lowel.tiggen.com/edu/color_temperature_and_rendering_demystified.html.
2. Walt Disney Animation Studios. Hyperion [Електронний ресурс] / Walt Disney Animation Studios. – 2015. – Режим доступу до ресурсу: https://www.disneyanimation.com/technology/innovations/hyperion.
3. C. Eisenacher, G. Nichols, A. Selle and B. Burley, “Sorted deferred shading for production path tracing,” in EGSR '13, Aire-la-Ville, 2013, pp. 125-132.
4. Pixar Animation Studios. About RIS [Електронний ресурс] / Pixar Animation Studios. – 2015. – Режим доступу до ресурсу: https://renderman.pixar.com/resources/current/RenderMan/risOverview.html.
5. Otoy Inc. Brigade: real-time path tracing [Електронний ресурс] / Otoy Inc. – 2015. – Режим доступу до ресурсу: https://home.otoy.com/render/brigade/.
6. LuxRender Developer Community. LuxRender: GPL Physically Based Renderer [Електронний ресурс] / LuxRender Developer Community – Режим доступу до ресурсу: http://www.luxrender.net/en_GB/index.
7. Pharr M. “pbrt-v2” [Електронний ресурс] / M. Pharr, G. Humphreys – Режим доступу до ресурсу: https://github.com/mmp/pbrt-v2.
8. Pharr M. “pbrt-v3” [Електронний ресурс] / M. Pharr, G. Humphreys – Режим доступу до ресурсу: https://pbrt.org/scenes-v3.html.
9. Oren M. Generalization of Lambert’s Reflectance Model [Електронний ресурс] / M. Oren, S. K. Nayar // SIGGRAPH 94. – 1994. – Режим

доступу до ресурсу: http://www1.cs.columbia.edu/CAVE/publications/pdfs/Oren_SIGGRAPH94.pdf.

10. K. E. Torrance Theory for off-specular reflection from roughened surfaces / K. E. Torrance, E. M. Sparrow. // Journal of the Optical Society of America. – 1967. – №9. – С. 1105–1114.
11. Stephen M. Inverse rendering for computer graphics [Електронний ресурс] / Marschner Stephen – Режим доступу до ресурсу: https://www.graphics.cornell.edu/pubs/1998/Mar98.pdf.
12. Cornell Univ. Computer Graphics Dept. Reflectance Data: Cornell University Program of Computer Graphics [Електронний ресурс] / Cornell University – Режим доступу до ресурсу: http://www.graphics.cornell.edu/online/measurements/.
13. Turner W. An Improved Illumination Model for Shaded Display [Електронний ресурс] / Whitted Turner. – 2005. – Режим доступу до ресурсу: http://artis.imag.fr/Members/David.Roger/whitted.pdf.
14. Kajiya J. The Rendering Equation [Електронний ресурс] / James Kajiya – Режим доступу до ресурсу: http://www.cse.chalmers.se/edu/year/2011/course/TDA361/2007/rend_eq.pdf.
15. CIE Technical Report, Method of measuring and specifying colour rendering properties of light sources, Publ. CIE 13.3-1995, CIE Central Bureau Vienna, Austria
16. Bodrogi P. Colour rendering: past, present(2004), and future [Електронний ресурс] / Bodrogi P. – Режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.463.439&rep=rep1&type=pdf.
17. S. Watanabe, S. Kanamori, S. Ikeda, B. Raytchev, T. Tamaki and K. Kaneda, “Performance Improvement of Physically based spectral rendering using stochastic sampling,” in CCIW'13 Proceedings of the 4th

international conference on Computational Color Imaging, Chiba, 2013, pp. 184-198.

18. C. J. Bartleson, Colorimetry, in Optical Radiation Measurements, Vol. 2: Color Measurement, Eds. F. Grum and C. J. Bartleson, Academic Press, New York, 1980, pp. 33-148.
19. Kolb H. Simple Anatomy of the Retina [Електронний ресурс] / Helga Kolb – Режим доступу до ресурсу: http://webvision.med.utah.edu/book/part-i-foundations/simpleanatomy-of-the-retina/.
20. Kang H. Tristimulus Specification [Електронний ресурс] / Henry R. Kang – Режим доступу до ресурсу: https://www.spiedigitallibrary.org/ebooks/PM/Computational-Color-Technology/Chapter1/Tristimulus-Specification/10.1117/3.660835.ch1?SSO=1.
21. MacAdam D. Projective Transformations of I. C. I. Color Specifications / David L. MacAdam. // Journal of the Optical Society of America. – 1937. – №8. – С. 294–299.
22. Wyszecki G. Proposal for a New Color-Difference Formula / Wyszecki G. // Journal of the Optical Society of Amer. – 1963. – №11. – С. 1318–1319.
23. X-Rite Inc. A Guide to Understanding Color Communication [Електронний ресурс] / X-Rite Inc. – Режим доступу до ресурсу: https://www.xrite.com/documents/literature/en/l10-001_understand_color_en.pdf.
24. Lindbloom B. J. XYZ to Lab [Електронний ресурс] / B. J. Lindbloom – Режим доступу до ресурсу: http://www.brucelindbloom.com/index.html?Eqn_XYZ_to_Lab.html.
25. CIE Commission Internationale de l'Eclairage. CIE Standard Illuminants for Colorimetry [Електронний ресурс] / CIE Commission Internationale

de l'Eclairage – Режим доступу до ресурсу: http://www.cie.co.at/publ/abst/s005.html.

26. Spectral Distribution of Typical Daylight as a Function of Correlated Color Temperature / [G. Judd, D. L. MacAdam, G. Wyszecki та ін.]. // Journal of the Optical Society of America. – 1964. – №8. – С. 1031–1040.
27. CIE Commission Internationale de l'Eclairage. Colorimetry [Електронний ресурс] / CIE Commission Internationale de l'Eclairage – Режим доступу до ресурсу: https://www.cdvplus.cz/file/3-publikace-cie15-2004/.
28. McCamy C. A Color-Rendition Chart / C. S. McCamy, H. Marcus, J. G. Davidson. // Journal of Applied Photographic Engineering. – 1976. – №3. – С. 95–99.
29. Pascale D. RGB coordinates of the Macbeth ColorChecker [Електронний ресурс] / Danny Pascale – Режим доступу до ресурсу: http://www.babelcolor.com.
30. CIE Commission Internationale de l'Eclairage, “A review of the chromatic adaptation transform,” Tech. Rep. CIE 160:2004, CIE Commission Internationale de l'Eclairage, Vienna, 2004.
31. Westland S. Chromatic-adaptation Transforms and Colour Appearance / S. Westland, C. Ripamonti // John Wiley & Sons. – 2004. – №6. – С. 86–88.
32. M. Pharr and G. Humphreys. Physically Based Rendering: From Theory to Implementation. – 2ed. – Morgan Kaufmann Publisher Inc., San Francisco, CA, USA, 2010, ch. 5, pp. 261-297.
33. M. Pharr and G. Humphreys. Physically Based Rendering: From Theory to Implementation. – 2ed. – Morgan Kaufmann Publisher Inc., San Francisco, CA, USA, 2010, ch. 5, pp. 760-770.
34. Lafortune E. Bi-Directional Path Tracing / E. P. Lafortune, Y. D. Willems. // Compugraphics. – 1993. – С. 145–153.

35. Jang H. Indoor Lighting [Електронний ресурс] / Hin Jang – Режим доступу до ресурсу: https://hinjang.com/articles/03.html.
36. Phong B. Illumination of Computer-Generated Images / Bui Tuong Phong. // Communications of the ACM. – 1975. – №6. – С. 311–317.
37. Blinn J. Models of light reflection for computer synthesized pictures [Електронний ресурс] / James F. Blinn // SIGGRAPH '77. – 1977. – Режим доступу до ресурсу: http://cs.uns.edu.ar/cg/clasespdf/p192-blinn.pdf.
38. Pharr M. Reflection Models [Електронний ресурс] / M. Pharr, G. Humphreys. – 2018. – Режим доступу до ресурсу: http://www.pbr-book.org/3ed-2018/Reflection_Models.html
39. Dirac P. Representations / Paul Dirac // The Principles of Quantum Mechanics / Paul Dirac. – Oxford: Clarendon Press, 1958. – (4-th edition). – С. 58.
40. Laplante P. Designing Software / Phillip Laplante // What Every Engineer Should Know about Software Engineering (What Every Engineer Should Know) / Phillip Laplante. – Boca Raton: CRC Press, 2007. – С. 85–86.
41. Vandevenne L. LodePNG [Електронний ресурс] / L. Vandevenne – Режим доступу до ресурсу: http://lodev.org/lodepng/.
42. Apple Inc. Cocoa Application Layer [Електронний ресурс] / Apple Inc. –
Режим доступу до ресурсу: https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/OSX_Technology_Overview/CocoaApplicationLayer/CocoaApplicationLayer.html.
43. Apple Inc. Cocoa Touch Layer [Електронний ресурс] / Apple Inc. – Режим
доступу до ресурсу: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html.

44. Apple Inc. Grand Central Dispatch (GCD) [Електронний ресурс] / Apple Inc. – Режим доступу до ресурсу: https://developer.apple.com/library/ios/documentation/Performance/Reference/GCD_libdispatch_Ref/.
45. Microsoft Corporation. Asynchronous programming in C++ [Електронний ресурс] / Microsoft Corporation – Режим доступу до ресурсу: https://msdn.microsoft.com/windows/uwp/threadingasync/asynchronous-programming-in-cpp-universal-windows-platform-apps.
46. Travis CI, GmbH. Travis CI: builds apps with confidence [Електронний ресурс] / Travis CI, GmbH – Режим доступу до ресурсу: https://travis-ci.com.
47. Appveyor Systems Inc. AppVeyor: Continuos Delivery service for Windows [Електронний ресурс] / Appveyor Systems Inc. – Режим доступу до ресурсу: https://www.appveyor.com.
48. Hoxley J. D3DBook: (Lighting) Oren-Nayar [Електронний ресурс] / Jack Hoxley – Режим доступу до ресурсу: http://content.gpwiki.org/D3DBook:(Lighting)_Oren-Nayar.
49. Pharr M. Reflection Models [Електронний ресурс] / M. Pharr, G. Humphreys. – 2018. – Режим доступу до ресурсу: http://www.pbr-book.org/3ed-2018/Monte_Carlo_Integration.html
50. E. Catmull. A subdivision algorithm for computer display of curved surfaces. Phd Thesis, Utah Univ., 1974.
51. Greene N. Environment mapping and other applications of world projections / Ned Greene. // IEEE Computer Graphics and Applications. – 1986. – №11. – С. 21–29.
52. Ken P. An image synthesizer [Електронний ресурс] / Perlin Ken // SIGGRAPH. – 1985. – Режим доступу до ресурсу: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.220.2248&rep=rep1&type=pdf.

53. Ken P. Improving noise / Perlin Ken. // SIGGRAPH. – 2002. – С. 681–682.
54. Blinn J. Simulation of wrinkled surfaces [Електронний ресурс] / James F. Blinn – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/wp-content/uploads/1978/01/p286-blinn.pdf.
55. Stanford computer graphics laboratory. The Stanford 3D Scanning Repository [Електронний ресурс] / Stanford computer graphics laboratory – Режим доступу до ресурсу: http://graphics.stanford.edu/data/3Dscanrep/.
56. Visual Computing Laboratory CNR. MeshLab [Електронний ресурс] / Visual Computing Laboratory CNR – Режим доступу до ресурсу: http://meshlab.sourceforge.net.
57. Scratchapixel. Ray-Box Intersection [Електронний ресурс] / Scratchapixel – Режим доступу до ресурсу: http://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracerrendering-simple-shapes/ray-box-intersection.
58. CIE Commission Internationale de l'Eclairage, “International Lighting Vocabulary,” Tech. Rep. 17.4-1987, CIE Commission Internationale de l'Eclairage, Vienna, 1987.
59. Beason K. smallpt: Global Illuminantion in 99 lines of C++ [Електронний ресурс] / Kevin Beason – Режим доступу до ресурсу: http://www.kevinbeason.com/smallpt/.
60. Ward G. Measuring and modeling anisotropic reflection [Електронний ресурс] / Gregory J. Ward // SIGGRAPH. – 1992. – Режим доступу до ресурсу: https://cseweb.ucsd.edu/~ravir/6998/papers/p265-ward.pdf.
61. Ashikhmin M. An Anisotropic Phong BRDF Model / M. Ashikhmin, P. Shirley. // Journal of Graphics Tools. – 2000. – №2. – С. 25–32.

ДОДАТОК 1. СПЕКТРАЛЬНІ КООРДИНАТИ

Вектор у тривимірному просторі, який описується декартовою системою координат, визначається як $v = (v_x, v_y, v_z)$, причому кожна компонента пов'язана з конкретним виміром. У розрахунку BRDF корисним є опис вектора також у сферичній системі координат.

У такій системі вектор задається з величиною ρ і парою кутів θ та ϕ. Перший – це полярний кут, виміряний від напрямку зеніту. Другий кут – це кут азимута проекції вектора на дотичну площини до початку координат і він є ортогональним до зеніту, виміряний від фіксованого опорного напрямку на площині.

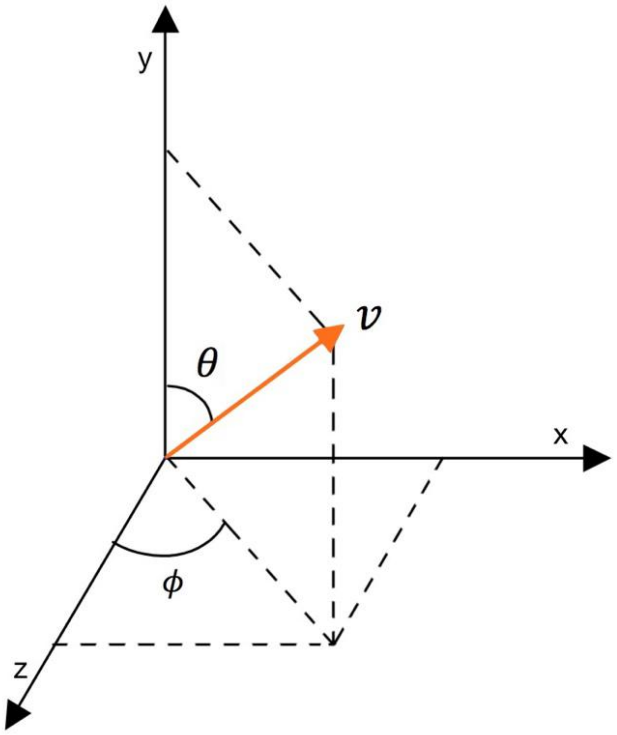

Рисунок 1 – Сферична система координат

Формули, які використовуються для перетворення системи координат з декартової у сферичну мають наступний вигляд:

$$\phi = \arctan\frac{v_z}{v_x} \quad (1)$$

$$\theta = \arccos v_y \quad (2)$$

Обернене перетворення, яке також припускає наявність нормалізованого вектора, зі сферичної системи координат в декартову можливе, використовуючи такі рівняння:

$$v_x = \cos\phi \sin\theta \tag{4}$$

$$v_y = \cos\theta \tag{5}$$

$$v_z = \sin\phi \sin\theta \tag{6}$$

ДОДАТОК 2. TTFD СЦЕНА. ФАЙЛ ФОРМАТУ XML

TTFD використовує свій власний формат XML для визначення сцени. Кожен з тегів, що містяться у цих XML-файлах, пов’язаний з: компонентом/об’єктом сцени, властивістю конфігурації TTFD або властивістю конфігурації певної сцени. Нижче наведено список цих тегів та їх атрибутів. Цей розділ може розглядатися як посібник користувача.

B.1 scene

Тег сцена є кореневим вузлом xml-файлу TTFD. Він містить усі інші теги, які визначають сцени TTFD.

Таблиця 4 – Список атрибутів тагу scene

Атрибут	Значення	Опис
Type	String: RGB or Spectral	Визначає тип сцени. Якщо не визначено, то по замовчуванню - спектральна

B.2 camera

Таг камера містить всі таки, які визначають інформацію, яка необхідна для створення камери у TTFD. Цей таг не має атрибутів і він мусить бути тільки один для сцени, описаній в XML файлі.

B.3 viewReferencePoint

Таг viewReferencePoint визначає точку огляду для камери. Він має міститися у тегу camera.

Таблиця 5 – Список атрибутів тагу viewReferencePoint

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.4 lookAtPoint

Таг lookAtPoint визначає точку, на яку направлена камера. Це має міститися в тагу camera.

Таблиця 6 – Список атрибутів тагу lookAtPoint

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.5 viewPlane

Таг viewPlane визначає площину зображення. Він має знаходитись у тагу camera. У момент написання лише відстань може бути налаштована у сцені XML файлу.

Таблиця 7 – Список атрибутів тагу viewPlane

Атрибут	Значення	Опис
d	float	Відстань до площини зображення

B.6 light

Таг light визначає джерело світла у сцені. В даний момент TTFD підтримує лише одне джерело світла для сцени.

Таблиця 8 – Список атрибутів тагу light

Атрибут	Значення	Опис
type	String: area or point	Визначає тип світла, яке використовується
spectrum	String: illuminant spectrum name	Визначає джерело світла у сцені. Використовується лише у спектральних сценах
brightness	Float	Яскравість світла (трасування шляху)

B.7 tracerModelType

Таг tracerModelType визначає тип трасувальника, який буде використовуватися для рендеринга сцени. Цей таг може містити одне з трьох значень, які відносяться до моделей TTFD.

- TracerRGBModel;
- TracerSpectrumModel;
- PathTracerModel.

Таблиця 9 – Список атрибутів тагу tracerModelType

Атрибут	Значення	Опис
numberOfSamples	float	Кількість взятих зразків на піксель (трасування шляху)

B.8 shadingModelType

Таг shadingModelType визначає тип моделі затінення, який використовується під час рендеринга сцени. Цей таг може мати одне з трьох значень, які відносяться до моделей, реалізованих у TTFD.

- WhittedShadingModel;
- WhittedBRDFShadingModel;
- PathTracingBRDFShadingModel.

B.9 cubeMappingSkybox

Таг cubeMappingSkybox визначає скайбокс на нескінченній відсатін, використовуючи техніку мапування текстури куба для RGB сцен. Цей тег може приймати значення TRUE або FALSE. Якщо не визначено, то значення по замовчування false і текстура куба не буде завантажена.

Текстура, необхідна для створення скайбоксу, мусить бути розташована у папці зі сценою з іменем скайбоксу, який вона відображає.

B.10 empiricalLighting

Таг empiricalLighting визначає емпіричну модель світла.

Цей аг має сенс лише для RGB сцен, оскільки спектральна сцена використовує фізично обґрунтовані BRDF моделі. Цей таг може приймати лише два значення:

- Phong;
- BlinnPhong.

B.11 objects

Таг objects містить всі таги об’єктів, які представляють їх визначення. Кожна сцена повинна включати таг objects.

B.12 object

Таг object визначає об’єкти, які додаються в сцену.

Таблиця 10 – Список атрибутів тагу object

Атрибут	Значення	Опис
type	String: sphere, squere and polygonal shape.	Визначає тип об’єкту.
skyboxSide	String: bottom, left, right, back, front and top.	Визначає об’єкти як сторону skybox-у. Корисно тільки тоді, коли об’єкти квадратні.

B.13 origin

Таг origin використовується для визначення сферичних об’єктів і тих, що світяться.

Таблиця 11 – Список атрибутів тагу origin

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.14 radius

Таг radius використовується для визначення радіусу сферичних об’єктів і тих, що світяться.

Таблиця 12 – Список атрибутів тагу lookAtPoint

Атрибут	Значення	Опис
r	Float	Визначає радіус

B.15 color

Таг color визначає RGB колір світла. Цей таг корисний лише в RGB сцені і запиється в таг light.

Таблиця 13 – Список атрибутів тагу color

Атрибут	Значення	Опис
r	Float	Компонента червонного кольору
g	Float	Компонента зеленого кольору
b	Float	Компонента блакитного кольору

B.16 vertex

Таг vertex визначає вершину квадратної або багатокутної форми об’єкта. Цей таг корисний лише в межах тагу object, який визначається за допомогою значень типу squere або polygonalshape.

Таблиця 14 – Список атрибутів тагу vertex

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.17 pointOnPlane

Таг pointOnPlane визначає точку на площині, яка пов’язана з квадратної або багатокутної форми об’єктом. Цей так корисний тільки

всередині тагу object з визначеними значеннями типу squere та polygonalshape.

Таблиця 15 – Список атрибутів тагу origin

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.18 material

Таг material визначає зовнішній вигляд об’єкта. Це корисно тільки у випадку, якщо визначається всередині тагу object.

Таблиця 16 – Список атрибутів тагу pointOnPlane

Атрибут	Значення	Опис
type	String with type of material. RGB scene: • Jade • Bronze • Violet • Green • Red • mediumGray • lightGray • matte • matteTextured • chrome • silve • glass • glasswater • rubyBumpMapped	Визначає тип матеріалу, який відноситься до об’єкту

Продовження таблиці 16

	• flameMarble • blueTurbulance Spectral scene: • emissive • matteLambertian • matteOrenNayar • measured • plastic • glass • mirror	
textureName	String	Визначає використовувану текстуру. Текстури підтримуються лише в RGB сцені
scale	Float	Визначає шкалу, яка використовується з rubyBumpMapped матеріалом
spectrum	String	Визначає спектр, який використовується з matte lambertian або matte Oren Nayar спектральним матеріалом
spectrumDiffuse	String	Визначає дифузний спектр, який використовується з пластиковим спектральним матеріалом
spectrumSpecular	String	Визначає спектр, який

Продовження таблиці 16

		використовується пластиковим спектральним матеріалом
degree	float	Визначає ступінь, який використовується з matte Oren Nayar
name	String	Визначає назву виміряної BRDF для вимірюваного матеріалу
interpolated	bool	Визначає чи потрібно інтерполювати дані виміряної BRDF

B.19 mesh

Таг mesh використовується для визначення OBJ сітки для відтворення у сцені.

Сітка завжди повинна містити aabb таг, щоб визначити осі, які обмежують обмежуючий бокс (коробка, простір), які використовуються для прискорення тесту перетину і тег objFile для отримання файлу OBJ, який містить модель. Цей тег не має атрибутів.

B.20 objFile

Тег objFile використовується для визначення OBJ моделі, яка використовується для побудови сітки.

Таблиця 17 – Список атрибутів тагу objFile

Атрибут	**Значення**	**Опис**
Name	String	Визначає ім’я сітки OBJ файлу, який завантажується
backFaceCulling	Bool	Полігони з .OBJ файлу зникли/ні

B.22 min

Таг min використовується всередині тагу aabb для визначення мінімальної міри (протяжності, відстані) AABB.

Таблиця 18 – Список атрибутів тагу origin

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.23 max

Таг max використовується всередині aabb тагу для визначення максимальної міри AABB.

Таблиця 19 – Список атрибутів тагу origin

Атрибут	Значення	Опис
x	Float	Координата *x*
y	Float	Координата *y*
z	Float	Координата *z*

B.24 RGB сцена XML приклад

Наступний фрагмент показує приклад RGB сцени у XML файлі.

```
<scene type="RGB">
  <camera>
        <viewReferencePoint x="215" y="250" z="490" />
        <lookAtPoint x="250" y="250" z="250" />
        <viewPlane d="240" />
  </camera>
  <light type="area">
        <origin x="0" y="300" z="400" />
        <radius r="60" />
        <color r="255" g="255" b="255" />
  </light>
  <cubeMappingSkybox>TRUE</cubeMappingSkybox>
```

```
    <tracerModelType>TracerRGBModel</tracerModelType>
    <shadingModelType>WhittedShadingModel</shadingModelType>
    <empiricalLighting>Phong</empiricalLighting>
    <objects>
        <object type="sphere">
            <origin x="300" y="300" z="250" />
            <radius r="90" />
            <material type="glass" />
        </object>
        <object type="sphere">
            <origin x="230" y="380" z="50" />
            <radius r="100" />
            <material type="jade" />
        </object>
        <object type="sphere">
            <origin x="110" y="150" z="150" />
            <radius r="90" />
            <material type="bronze" />
        </object>
        <object type="sphere">
            <origin x="340" y="130" z="250" />
            <radius r="60" />
            <material type="rubyBumpMapped" scale="4.0" />
        </object>
    </objects>
</scene>
```

B.25 Спектральна сцена XML приклад

Наступний фрагмент показує приклад спектральної сцени у XML файлі. Сцена використовує трасування шляху як трасувальну модель і показу майже всі доступні таги і атрибути для спектрального рендеринга.

```
<scene type="Spectral">
    <camera>
        <viewReferencePoint x="512" y="250" z="1023" />
        <lookAtPoint x="512" y="250" z="512" />
        <viewPlane d="410" />
    </camera>
```

```
<light type="area" brightness="50" spectrum="d65">
        <origin x="500" y="650" z="-990" />
        <radius r="1000" />
</light>
<tracerModelType>
        PathTracerModel
</tracerModelType>
<shadingModelType numberOfSamples="40">
        PathTracingBRDFShadingModel
</shadingModelType>
<objects>
        <object type="polygonalshape">
                <vertex x="0" y="0" z="0" />
                <vertex x="0" y="0" z="1024" />
                <vertex x="1024" y="0" z="1024" />
                <vertex x="1024" y="0" z="0" />
                <pointOnPlane x="512" y="0" z="512" />
                <material type="matteLambertian" spectrum="neutral8" />
        </object>
        <object type="polygonalshape">
                <vertex x="0" y="0" z="0" />
                <vertex x="0" y="1024" z="0" />
                <vertex x="0" y="1024" z="1024" />
                <vertex x="0" y="0" z="1024" />
                <pointOnPlane x="0" y="512" z="512" />
                <material type="matteLambertian" spectrum="red" />
        </object>
        <object type="polygonalshape">
                <vertex x="1024" y="0" z="0" />
                <vertex x="1024" y="0" z="1024" />
                <vertex x="1024" y="1024" z="1024" />
                <vertex x="1024" y="1024" z="0" />
                <pointOnPlane x="1024" y="512" z="512" />
                <material type="matteLambertian" spectrum="green" />
        </object>
        <object type="polygonalshape">
                <vertex x="0" y="0" z="0" />
                <vertex x="1024" y="0" z="0" />
```

```
        <vertex x="1024" y="1024" z="0" />
        <vertex x="0" y="1024" z="0" />
        <pointOnPlane x="512" y="512" z="0" />
        <material type="matteLambertian" spectrum="white" />
</object>
<object type="polygonalshape">
        <vertex x="0" y="1024" z="1024" />
        <vertex x="0" y="1024" z="0" />
        <vertex x="1024" y="1024" z="0" />
        <vertex x="1024" y="1024" z="1024" />
        <pointOnPlane x="512" y="1024" z="512" />
        <material type="matteLambertian" spectrum="neutral8" />
</object>
        <object type="polygonalshape">
        <vertex x="0" y="0" z="1024" />
        <vertex x="0" y="1024" z="1024" />
        <vertex x="1024" y="1024" z="1024" />
        <vertex x="1024" y="0" z="1024" />
        <pointOnPlane x="512" y="512" z="1024" />
        <material type="matteLambertian" spectrum="white" />
</object>
<object type="sphere">
        <origin x="760" y="100" z="280" />
        <radius r="100" />
        <material type="matteOrenNayar"
        spectrum="orange"
        degree="50"/>
</object>
<object type="sphere">
        <origin x="540" y="90" z="450" />
        <radius r="90" />
        <material type="glass" />
</object>
<object type="sphere">
        <origin x="313" y="70" z="350" />
        <radius r="70" />
        <material type="glass" />
</object>
```

```
            <object type="sphere">
                <origin x="200" y="90" z="150" />
                <radius r="90" />
                <material type="matteLambertian"
                spectrum="purplishBlue" />
            </object>
    </objects>
</scene>
```